Commencé le	samedi 21 novembre 2020, 15:35
État	Terminé
Terminé le	samedi 21 novembre 2020, 20:07
Temps mis	4 heures 32 min
Points	12,50/15,00
Note	**16,67** sur 20,00 (**83**%)

Description

Quiz concernant : conclusion module 9, lectures module 10, résultats TP 5.

Votre note est disponible immédiatement à la fin du quiz, mais la correction est disponible uniquement après la fermeture du test.

Les questions à choix multiples disposent de réponses fausses à points négatifs.

Question **1**

Terminer

Note de 1,00 sur 1,00

Détection de pannes

- [x] a. Pour détecter qu'un participant dans un système réparti est en panne, il faut soit s'apercevoir qu'un message attendu n'arrive pas (détection passive), soit envoyer une requête pour sonder le participant (détection active).
- [ ] b. Il est utile de détecter si une panne est au niveau du réseau ou d'un noeud. Ceci est très facile à réaliser, quelle que soit la topologie du réseau.
- [ ] c. Sur certains systèmes, on ajoute un réseau secondaire pour vérifier si la panne est au niveau du réseau. Si un noeud ne répond pas sur le réseau secondaire, alors on peut être absolument certain, quelles que soient les circonstances, que le problème n'est pas au niveau du réseau mais bien au niveau du noeud.
- [x] d. Si le système et le réseau peuvent être lents, il est difficile d'établir un seuil au-delà duquel on considère que ne pas recevoir une réponse permet de conclure qu'un participant est en panne. On peut soit attendre trop longtemps et avoir une détection trop longue, soit avoir un faux positif (détecter une panne alors que le système est fonctionnel mais lent).

Question **2**

Terminer

Note de 0,00 sur 1,00

Interblocages

- a. Pour éviter les interblocages, il faut idéalement avoir un ordre strict de verrouillage. Ceci empêche la formation d'une dépendance circulaire entre les verrous et les transactions qui attendent après les verrous.
- b. La seule manière possible de se sortir d'un interblocage est de construire un graphe de dépendance et de vérifier s'il existe un cycle dans ce graphe.
- c. Le problème des interblocages n'existe pas vraiment. Il est très facile et il n'y a pas d'inconvénient à simplement demander tous les verrous d'une transaction en bloc. Si l'ensemble des verrous est obtenu, la transaction ne peut bloquer. Si l'ensemble est refusé, aucun verrou n'est pris et il faut se réessayer mais rien n'est bloqué pour les autres transactions.
- d. En pratique, par exemple avec des clients humains qui font interactivement du commerce en ligne, il est difficile d'assurer un ordre de verrouillage des données, et il faut alors avoir une manière de détecter les interblocages et d'annuler certaines transactions.

Question **3**

Terminer

Note de 1,00 sur 1,00

Modèles de cohérence

- a. La cohérence est très facile à obtenir, il suffit que chaque client envoie ses écritures à tous les serveurs fonctionnels au moment de l'écriture.
- b. La cohérence causale demande que chaque message vienne avec une copie de tous les messages dont il dépend.
- c. La cohérence stricte est lorsque la mise à jour apparaît en même temps sur toutes les copies; on ne peut lire en même temps deux copies différentes sur deux réplicats.
- d. La cohérence causale est un peu moins contraignante que la cohérence séquentielle.

Question **4**

Terminer

Note de 1,00 sur 1,00

Transactions réparties

- a. Le problème avec le protocole de fin de transaction atomique à 2 phases est que, si un des participants est en panne, tous les participants seront bloqués indéfiniment, sans possibilité que l'erreur soit détectée et la transaction annulée.
- b. Le protocole de fin de transaction atomique à 2 phases est aussi applicable en grande partie pour des transactions avec des serveurs répliqués.
- c. Pour une transaction répartie, il faut s'assurer de l'atomicité à travers plusieurs processus par l'envoi de messages. Ceci peut se faire par un protocole de fin de transaction atomique à 2 phases, envoyer l'information à tous les participants et demander leur accord dans une première phase, et confirmer s'il y a lieu la transaction dans une deuxième phase. Cette manière de procéder est similaire à celle utilisée pour les messages de groupes atomiques.
- d. Il n'est pas possible d'avoir des transactions réparties, car les horloges ne peuvent jamais être parfaitement synchronisées et cela empêche de valider l'ordre entre les transactions sur chaque serveur.

Question **5**

Terminer

Note de 0,50 sur 1,00

Récupération en cas de panne

- a. Un des défis avec la méthode du journal est de libérer l'espace occupé par le journal. Une fois que les écritures des transactions acceptées ont été propagées à la copie maîtresse de la base de donnée, elles ne sont plus requises dans le journal et ces entrées pourraient être libérées.
- b. Une seule écriture atomique sur disque est requise pour sauvegarder toutes les variables modifiées par une transaction, ainsi que la confirmation que la transaction est acceptée. Il n'y a pas de raison de prévoir faire cette opération en plusieurs écritures sur disque, ce qui serait plus long et moins efficace.
- c. La méthode du journal est la seule possible pour assurer la persistence des données en cas de redémarrage du système.
- d. Toutes les écritures associées à une transaction peuvent être sauvegardées dans un journal. On peut ensuite écrire de manière atomique l'information disant que la transaction est acceptée. Il est alors possible d'envoyer le message d'acceptation de la transaction au client, sachant que même en cas de redémarrage, le journal contient l'information sur la transaction acceptée.

Question **6**

Terminer

Note de 0,00 sur 1,00

La gestion de la composition des groupes

- a. Les messages atomiques sont très robustes, le fait d'apprendre ou non qu'un noeud est en panne, et ne fait pas partie du groupe, ne change rien au bon fonctionnement des envois atomiques dans un groupe.
- b. La composition des groupes peut être maintenue en réparti auprès de tous les participants dans le groupe. Une difficulté importante est de gérer les changements de groupe et de synchroniser ces changements auprès de tous les participants. Ceci peut être fait avec des messages atomiques (tous le reçoivent ou aucun) et ordonnés (les changements de la composition du groupe arrivent dans le même ordre par rapport aux messages de groupe pour tous les participants).
- c. Pour l'envoi de messages de groupe, il est important de connaître la composition du groupe. Un serveur central peut maintenir la composition des groupes et s'occuper de relayer les messages aux membres du groupe. Cependant, dans cette organisation, tout repose sur le serveur central et il ne faut pas qu'il tombe en panne.
- d. L'envoi d'un message atomique requiert (n x n) / 2 messages, puisque chaque participant doit vérifier auprès de chaque autre participant qu'il a bien reçu le message et est prêt à le livrer à l'application.

Question **7**

Terminer

Note de 1,00 sur 1,00

Paxos

- a. Une difficulté importante, pour les algorithmes qui veulent établir un consensus, est de pouvoir fonctionner même si certains participants peuvent tomber en panne et revenir, pendant qu'on essaie d'établir un consensus. L'algorithme de l'élection hiérarchique est simple et fonctionne bien tant qu'il n'y a pas trop de participants qui tombe en panne pendant l'élection.
- b. L'algorithme de Paxos, pour établir un consensus, utilise plusieurs fonctions: proposeur, accepteur, apprenant. Ceci donne une flexibilité pour aider à faire converger le consensus, même lorsque des pannes surviennent.
- c. Dans l'algorithme de Paxos, un accepteur accepte la première proposition reçue et ignore les suivantes, reçues d'autres proposeurs. Ceci assure une cohérence forte et évite l'anarchie.
- d. L'algorithme de Paxos n'est pratiquement jamais utilisé dans les gros systèmes, ce qui est surprenant car il s'agit d'un des algorithmes les plus simples disponibles.

Question **8**

Terminer

Note de 1,00 sur 1,00

Contrôle de la concurrence par prise de verrou

- a. Les verrous de lecture, contrairement aux verrous d'écriture, peuvent être relâchés dès que la lecture a été effectuée, même si la transaction n'est pas encore commise.
- b. Pour un contrôle de la concurrence par les verrous, en pratique les verrous sont pris au fur et à mesure que les variables sont accédées, et ne sont pas relâchés avant que la transaction ne soit commise.
- c. Un peu comme le serveur central d'exclusion versus l'exclusion en réparti, il est beaucoup plus efficace d'avoir un seul verrou pour toute la base de données, plutôt que d'avoir un verrou pour chaque variable. Cela aide la mise à l'échelle et le parallélisme.
- d. Il est possible d'avoir des verrous partagés en lecture. Ceux-ci servent à s'assurer qu'il n'y a pas d'écriture tant que le verrou de lecture est pris.

Question **9**

Terminer

Note de 1,00 sur 1,00

Récupération vers l'arrière et vers l'avant

- a. La récupération vers l'avant est un concept théorique impossible à réaliser en pratique. Il n'est pas possible de deviner et ensuite réparer tous les dégats possibles qui peuvent être causés par des pannes sur un système.
- b. Lorsqu'un ordinateur plante, alors qu'il était en train d'écrire sur disque, son système de fichiers peut être corrompu en raison des modifications interrompues. Par exemple, des blocs libérés par un fichier effacé pourraient ne pas encore avoir été ajoutés à la liste des blocs libres. La récupération vers l'arrière repartirait de la dernière copie de sauvegarde disponible. La récupération vers l'avant pourrait faire une vérification de la cohérence des structures de données du système de fichiers et découvrir et replacer les blocs libres manquants de la liste.
- c. Pour faire la récupération vers l'arrière, on peut prendre un cliché de l'état du système de temps en temps. On peut alors revenir vers le dernier état sauvé en cas de panne. Le travail depuis le dernier état sauvé jusqu'à la panne peut être perdu cependant.
- d. Il est possible de prendre un cliché d'un seul système mais il est impossible de prendre un cliché cohérent de l'état courant sur un système réparti avec plusieurs noeuds qui communiquent par message.

Question **10**

Terminer

Note de 1,00 sur 1,00

Redondance temporelle ou physique

- a. La redondance temporelle n'est d'aucune utilité en informatique, puisque les ordinateurs donnent toujours les même réponses, pour le même programme et les mêmes entrées, même si l'exécution est répétée des millions de fois.
- b. Les circuits de mémoire, avec une redondance qui permet la détection et la correction d'erreur, sont une forme de redondance physique.
- c. Répéter une opération une seconde fois sur un système, si elle a échouée la première fois, est une forme de redondance temporelle.
- d. Avec la redondance physique, il suffit d'avoir deux circuits qui font les calculs en parallèle. Si les réponses diffèrent, il est facile de simplement retenir le résultat valide.

Question **11**

Terminer

Note de 1,00 sur 1,00

Scénarios de panne

- a. Si un client prend trop de temps pour compléter une transaction, par exemple en raison de la lenteur du réseau, et que le serveur abandonne la transaction, le client a toujours droit de refaire sa transaction dans les mêmes conditions et avec les mêmes valeurs, comme si elle n'avait jamais été abandonnée.
- b. Si un client se commet pour une transaction mais ne reçoit pas de réponse du serveur, il ne sait pas si la transaction est acceptée ou non. Il doit recontacter le serveur pour confirmer si la transaction a été acceptée, ou si elle disparu parce que le serveur a redémarré avant de la compléter.
- c. Si un client redémarre, il peut avoir oublié toute transaction qu'il avait initiée. Le serveur décidera d'annuler la transaction, après un certain délai sans nouvelles de ce client.
- d. Si un client se commet pour une transaction mais ne reçoit pas de réponse du serveur, il peut simplement assumer que la transaction a été abandonnée.

Question **12**

Terminer

Note de 1,00 sur 1,00

Transactions imbriquées

- [x] a. Les transactions imbriquées permettent d'avoir des transactions à l'intérieur des transactions. Toutefois, les transactions internes demeurent conditionnelles à la confirmation et à l'acceptation des transactions englobantes.
- [x] b. Les transactions au même niveau d'imbrication peuvent s'exécuter en parallèle.
- [ ] c. Pour savoir si une transaction est finalement commise, on compte le nombre d'annulations et d'acceptations des transactions dans la hiérarchie des transactions parentes. S'il y a au moins une acceptation et un nombre pair d'annulations (i.e. deux annulations s'annulent), cette transaction est commise, autrement elle est annulée.
- [ ] d. L'imbrication des transactions est un artifice de syntaxe. Il n'y a pas de lien de dépendance entre les transactions imbriquées les unes dans les autres.

Question **13**

Terminer

Note de 1,00 sur 1,00

Modèles de pannes

- [x] a. Une panne par omission (pas de réponse après un délai maximum) est plus difficile à détecter et gérer que de recevoir un code d'erreur explicite.
- [ ] b. Sur un système asynchrone, les pannes par omission sont plus rapides à détecter, car la réponse vient normalement immédiatement après la réception d'une requête, tandis que sur un système synchrone, la réponse attend au prochain intervalle.
- [ ] c. Il n'y a pas vraiment de différence entre une panne byzantine et une panne de réponse. Il n'y a pas d'intérêt à traiter les deux cas séparément.
- [x] d. Une panne de plantage (e.g. l'ordinateur redémarre) a l'avantage d'être plus facile à déceler qu'une panne de réponse (e.g. mauvaise valeur).

Question **14**

Terminer

Note de 1,00 sur 1,00

Contrôle optimiste de la concurrence

- a. La méthode de vérification optimiste de la cohérence vers l'arrière valide qu'aucune transaction concurrente terminée n'a écrit une variable lue par la transaction courante.
- b. Dans le contrôle optimiste de la concurrence, les opérations d'une transactions sont reçues jusqu'à l'annulation (on laisse tomber) ou la confirmation (commit). Lorsque la confirmation est reçue, une vérification de cohérence est faite et la transaction est acceptée ou refusée par le serveur.
- c. La méthode de vérification optimiste de la concurrence par estampille de temps est plus complexe et moins précise que les méthodes vers l'avant et vers l'arrière. Elle n'est donc d'aucun intérêt.
- d. La méthode de vérification optimiste de la cohérence vers l'avant valide que la transaction courante n'a pas lu une variable déjà écrite par une transaction concurrente non terminée.

Question **15**

Terminer

Note de 1,00 sur 1,00

Faute (fault), erreur (error) et panne (failure)

- a. Une panne peut causer une erreur, ce qui devient une faute du système.
- b. Une erreur de programmation, ou une mauvaise connexion dans un circuit est une faute.
- c. Dans le contexte de la fiabilité et disponibilité des systèmes, les termes faute (fault) et panne (failure) sont des synonymes.
- d. Une erreur dans un état interne d'un programme ne cause pas nécessairement d'erreur visible (panne) en sortie. De la même manière, une erreur de programmation peut ne pas causer d'erreur en sortie pour beaucoup de cas d'utilisation.

◄ Quiz semaine 11 du 09/11

Aller à...

Quiz semaine 13 du 23/11 ►