POLYTECHNIQUE MONTRÉAL

Département de génie informatique et génie logiciel

Cours INF8480: Systèmes répartis et infonuagique (Automne 2024)
3 crédits (3-1.5-4.5)

CORRIGÉ DU CONTRÔLE PÉRIODIQUE

DATE: Lundi le 28 octobre 2024

HEURE: 15h45 à 17h35

DUREE: 1H50

NOTE: Aucune documentation permise sauf un aide-mémoire, 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) De nombreux clients font chacun 10 requêtes par seconde vers un serveur. Chaque requête demande au serveur 4ms sur un coeur de CPU et en plus, dans 25% des cas, la lecture d'un disque pendant 20ms. Chaque serveur possède 4 coeurs de CPU et 6 disques. Les requêtes sont bien réparties entre les coeurs de CPU et entre les disques. Le service utilise plusieurs fils d'exécution afin de servir en parallèle les requêtes. i) Quel est le facteur limitant entre les coeurs de CPU et les disques? ii) Quel est le nombre maximal de clients possible avant que le service ne sature? iii) Combien de fils d'exécution devrait-on rouler sur le serveur au minimum pour utiliser le plus possible les ressources disponibles? **(2 points)**

Puisque chaque requête prend 4ms sur un coeur de CPU, chaque coeur peut servir 1000ms/s / 4ms/r = 250 requêtes / seconde. Le total est donc de 1000 requêtes / seconde pour les 4 coeurs de CPU. Chaque requête prend en moyenne 0.25 x 20ms = 5ms de disque. Chaque disque peut servir 1000ms/s / 5ms/r = 200 requêtes / seconde. Le total est donc de 1200 requêtes / seconde pour les 6 disques. i) Le facteur limitant est les coeurs de CPU. ii) Ils peuvent supporter 1000 requêtes / seconde soit 1000r/s / 10r/client-s = 100 clients. iii) Si chaque requête reste dans le système 4ms CPU + 5ms disque = 9ms en moyenne, cela demande 1000r/s x 1s/1000ms x 9ms-thread/r = 9 thread, soit 9 fils d'exécution.

b) Un service de fichiers prend 8ms pour servir une requête. Les requêtes arrivent selon un processus de Poisson et sont mises en file d'attente lorsque le serveur est déjà occupé par une requête. Les requêtes arrivent au rythme moyen de 100 / seconde. i) Quel est le taux d'utilisation U du serveur. ii) Calculez N, le nombre moyen de requêtes dans le système. iii) Quel est W le temps de réponse moyen en secondes pour ce cas? **(2 points)**

Le taux de service u est de 1000ms/s / 8ms/r = 125r/s et le taux d'utilisation U = l/u = 100r/s / 125r/s = 0.8. En conséquence, N = 0.8 / (1 - 0.8) = 4, et W = N/l = 4 / 100r/s = .04s ou 40ms.

c) Les téléphones intelligents peuvent exécuter de nombreuses applications. Il est important que chaque application ne puisse par défaut accéder que ses propres données. Quel mécanisme est utilisé à cette fin sur les systèmes Android? Sur les systèmes IOS? **(1 point)**

Dans un cas comme dans l'autre, chaque application installée a son propre répertoire de données. Dans le cas de Android, chaque application se voit attribuer un identificateur d'usager différent. Le mécanisme de protection des fichiers sur la base de l'identificateur d'usager, fourni par le système d'exploitation Linux, sert donc de mécanisme pour assurer la séparation des fichiers entre les applications. Sur IOS, toutes les applications sont exécutées avec le même identificateur d'usager. Cependant, chaque application se fait attribuer un répertoire différent dont le nom long et aléatoire sert de clé d'accès. Une application, qui ne connait pas le nom du répertoire d'une autre application, ne peut pas, en temps raisonnable, en deviner le nom.

Question 2 (5 points)

a) Un système d'appel de procédure à distance utilise une sémantique *au plus une fois* avec le protocole sans connexion UDP. i) Expliquez combien de paquets et lesquels sont échangés entre

le client et le serveur pour compléter un appel à distance, dans le cas simple où aucun paquet ne sera perdu, incluant ce qui est nécessaire pour libérer les ressources associées à cet appel dans le serveur. ii) Si le système d'appel à distance utilisait plutôt la sémantique *au moins une fois*, comment cela changerait-il? **(2 points)**

ii) Avec la sémantique au moins une fois, un paquet avec la requête, du client au serveur, et un paquet avec la réponse, du serveur au client, suffit. i) Pour la sémantique au plus une fois, le serveur mémorise le numéro de la requête et la réponse associée. Ainsi, si la requête est retransmise par le client, parce que le paquet de réponse a été perdu, la réponse mémorisée est retransmise. Cependant, avec cette sémantique, le client doit en plus envoyer un accusé de réception de la réponse, pour permettre au serveur de libérer l'espace pris par la réponse mémorisée. Cela fait donc 3 paquets au lieu de 2.

b) Un appel de procédure à distance doit contenir les arguments suivants: string prenom, string nom, string candidat, int id. Si les valeurs sont: "Louis-Jean", "Symphorien", "Pikachu Ninjago", "778899", combien d'octets seront requis pour encoder cette information avec: i) CORBA CDR (32 bits)? ii) Avec gRPC protobuf? **(2 points)**

Avec CORBA CDR, nous avons 4 octets pour la longueur du prénom et 12 octets (multiple de 4) pour les 10 lettres de "Louis-Jean", 4 octets pour la longueur du nom et 12 octets (multiple de 4) pour les 10 lettres de "Symphorien", 4 octets pour la longueur du candidat et 16 octets (multiple de 4) pour les 15 lettres de "Pikachu Ninjago", et 4 octets pour le id. i) Le total est donc de 4+12+4+12+4+16+4 = 56 octets. Pour le format protobuf, nous avons pour le prénom 1 octet type/champ, 1 octet longueur et 10 octets pour "Louis-Jean", pour le nom 1 octet type/champ, 1 octet longueur et 10 octets pour "Symphorien", pour le candidat 1 octet type/champ, 1 octet longueur et 15 octets pour "Pikachu Ninjago", et pour le id 1 octet type/champ et 3 octets pour la valeur 778899 (0b10111110001010010011, soit 20 bits qui entrent dans 3 octets varint de 7 bits). ii) Le total est de 1+1+10+1+1+10+1+1+15+1+3 = 45 octets.

c) Le protocole SOAP (Simple Object Activation Protocol), basé sur XML, est souvent utilisé pour les appels de procédures à distance sur la Toile (Web). Quels sont les avantages et inconvénients de SOAP comparé à un protocole d'appel de procédure à distance comme gRPC avec protobuf? **(1 point)**

Le protocole SOAP est entièrement textuel et facile à comprendre. Il est donc facile à programmer et à mettre au point. Cependant, il est beaucoup moins efficace et compact qu'un protocole binaire comme gRPC avec protobuf.

Question 3 (5 points)

a) Un processus serveur reçoit des requêtes de clients par le biais d'appels de méthode à distance. Pour chaque requête, un objet de type *requete_client* est créé et une référence réseau à cet objet est retournée au client de la requête. Le client cesse d'utiliser cette référence réseau environ 80 secondes plus tard. Deux stratégies sont possibles pour prévenir le serveur lorsque l'objet de type *requete_client* n'est plus utile et peut être libéré. La stratégie *notification* procède en envoyant un message pour se désenregistrer de cet objet réseau exporté. La seconde stratégie, *bail renouvelable*, précise une durée de vie par défaut de l'objet après laquelle l'objet est libéré, 120 secondes dans ce cas-ci, à moins qu'un renouvellement ne soit demandé. Si le processus serveur reçoit 1 requête par 10 secondes de chaque client, et sert 500 clients, combien d'objets de type *requete_client* sont présents simultanément dans le processus serveur en moyenne: i) avec la stratégie *notification*, ii) avec la stratégie *bail renouvelable*? **(2 points)**

Le temps moyen qu'un objet reste alloué pour une requête avec la première stratégie est de 80s. Avec la seconde stratégie, ce temps est de 120s. Le taux d'arrivée des requêtes est de 1r/10s par client x 500 clients = 50 r/s. i) Le nombre total d'objets alloués simultanément pour la première stratégie est donc de 80s x 50 r/s = 4000. ii) Avec la seconde stratégie, le nombre total d'objets alloués simultanément est de 120s x 50 r/s = 6000.

b) Une machine virtuelle tourne sur un noeud physique Foo. L'image de cette machine virtuelle contient 8 000 000 pages. On veut migrer cette machine virtuelle vers un noeud Bar à travers un lien réseau qui permet d'envoyer 25 000 pages par seconde. Au premier tour, on copie l'ensemble des pages, aux tours subséquents, on copie les pages modifiées depuis le tour précédent. Pendant son exécution, la machine virtuelle modifie 5000 pages par seconde. La migration se fait d'abord en copiant les pages sans arrêter l'exécution puis, lorsqu'il reste peu de pages modifiées, l'exécution est arrêtée le temps de copier les pages restantes. Cette phase d'arrêt ne doit pas durer plus de 0.15 seconde. i) Combien de temps durera la migration au total? ii) Combien de temps est-ce que l'exécution sera arrêtée? **(2 points)**

Lors du premier tour, les 8 000 000 pages seront copiées en 8 000 000 / 25 000 = 320s. Pendant ces 320s, nous aurons 320s x 5000p/s = 1 600 000 pages modifiées. Au second tour, ces pages sont copiées en 1 600 000 / 25 000 = 64s, temps pendant lequel 64s x 5000p/s = 320 000 pages sont modifiées. Au troisième tour, les pages sont copiées en 320 000 / 25 000 = 12.8s, et 12.8s x 5000p/s = 64 000 pages sont modifiées. Au quatrième tour, les pages sont copiées en 64 000 / 25 000 = 2.56s, et 2.56 x 5000 = 12 800 pages sont modifiées. Au cinquième tour, les pages sont copiées en 12 800 / 25 000 = 0.512s, et 0.512 x 5000 = 2560 pages sont modifiées. Rendu là, l'exécution est interrompue, les pages restantes prenant moins que 0.15s à copier, et les 2560 pages restantes sont copiées en 2560 / 25 000 = 0.1024s. i) Le temps total pour la migration est donc de 320s + 64s + 12.8s + 2.56s + 0.512s + 0.1024s = 399.9744s. ii) L'exécution est arrêtée pendant le dernier tour qui dure 0.1024s.

c) La fonction KSM (Kernel Same page Merging) dans le noyau du système d'exploitation Linux est un mécanisme qui est présenté comme pouvant améliorer l'efficacité des machines virtuelles. Que fait cette fonction? A quoi sert-elle? Pourquoi est-ce particulièrement utile en présence de machines virtuelles? **(1 point)**

Avec KSM, une tâche en arrière plan vérifie si deux pages ou plus, en lecture seulement, ont exactement le même contenu. Si c'est le cas, ces pages sont fusionnées tant qu'elles ne sont pas modifiées. Cela permet de réduire les besoins en mémoire. Ceci est particulièrement utile pour les machines virtuelles, puisqu'elles utilisent souvent le même contenu (même version d'applications et de librairies dans chacune des machines virtuelles, par exemple).

Question 4 (5 points)

a) Un réseau de clients est servi par 3 serveurs CODA répliqués. Chaque client ouvre en moyenne 5 fichiers par seconde et en ferme autant. Lors de l'ouverture, le fichier n'est pas présent localement dans 40% des cas et doit être lu à partir d'un des trois serveur. Lors de la fermeture, le fichier a été modifié dans 20% des cas et doit alors être écrit sur chacun des 3 serveurs. Chaque requête de lecture prend 4ms de coeur de CPU sur un serveur et en plus, dans 30% des cas, une lecture d'un disque en 15ms. Chaque écriture prend sur chaque serveur 8ms de coeur de CPU et 20ms de temps d'un disque. Chaque serveur possède 8 coeurs de CPU et 16 disques. Si les requêtes sont bien réparties entre les serveurs, les coeurs de CPU et les disques, et le service utilise plusieurs fils d'exécution afin de servir en parallèle les requêtes, i) quel sera le facteur limitant entre les coeurs de CPU et les disques? ii) Quel est le nombre maximal de clients possible avant que le service ne sature? **(2 points)**

Chaque client ouvre 5 fichiers par seconde, ce qui demande: 5 x 0.4 = 2 lectures d'un des serveur, soit 0.666666 lecture par serveur. Chaque client ferme 5 fichiers par seconde, ce qui demande 5 x 0.2 = 1 écriture par serveur. Un client demande donc par serveur par seconde 0.666666 lecture x 4ms + 1 écriture x 8ms = 10.666666ms de coeur de CPU, et 0.666666 x 0.30 x 15ms + 1 x 20ms = 23ms de disque par serveur. Les 8 coeurs de CPU peuvent soutenir 8 x 1000ms/s / 10.666666ms/client-s = 750 clients. Les 16 disques peuvent soutenir 16 x 1000ms/s / 23ms/client-s = 695.65 clients. i) Le facteur limitant est les disques et ii) le système peut servir jusqu'à 695 clients avant de saturer.

b) Un volume sur un service GlusterFS est configuré pour être Réparti, Répliqué (Distributed Replicate). Les serveurs S1 et S2 offrent chacun une brique de disque de même dimension qui sont ainsi répliquées et forment le volume répliqué VR0. Les serveurs s3 et s4 font de même et forment le volume répliqué VR1. Finalement, le volume réparti répliqué VRR distribue ses fichiers entre VR0 et VR1. i) Si 10 fichiers, nommés `f0` à `f9`, sont placés sur ce volume et se distribuent équitablement entre les différentes localisations possibles, donnez une répartition possible de ces 10 fichiers sur les 4 serveurs. Indiquez bien quels fichiers se retrouvent sur la brique de chaque serveur. ii) Si on avait plutôt une configuration *Striped Replicate*, avec VR0 et VR1 qui sont joints pour former le volume VSR, donnez une répartition possible de ces 10 fichiers, ou morceaux de fichiers, sur les 4 serveurs. **(2 points)**

i) Les fichiers sur VRR vont se répartir entre VR0 et VR1. Ensuite, chaque fichier sur VR0 ou VR1 sera répliqué sur les 2 serveurs qui les composent. Ceci pourrait donner: S1 contient (f0, f2, f4, f6, f8), S2 (f0, f2, f4, f6, f8), S3 (f1, f3, f5, f7, f9), S4 (f1, f3, f5, f7, f9). ii) Pour le cas de VSR, chaque fichier pourrait être divisé en deux morceaux, par exemple f1 en f1/0 (fichier nommé f1 avec le contenu de la première moitié de f1 et un trou pour la seconde moitié) et f1/1 (fichier nommé f1 avec un trou pour la première moitié et ensuite le contenu de la seconde moitié

de f1). Nous aurions donc pour VSR: S1 contient (f0/0, f1/0, f2/0, f3/0, f4/0, f5/0, f6/0, f7/0, f8/0, f9/0), S2 (f0/0, f1/0, f2/0, f3/0, f4/0, f5/0, f6/0, f7/0, f8/0, f9/0), S3 (f0/1, f1/1, f2/1, f3/1, f4/1, f5/1, f6/1, f7/1, f8/1, f9/1), et S4 (f0/1, f1/1, f2/1, f3/1, f4/1, f5/1, f6/1, f7/1, f8/1, f9/1).

c) Pour plusieurs services de fichiers comme Lustre et Google File System, les serveurs de données sont en configuration active-active, alors que les serveurs de métadonnées sont en configuration active-passive. i) Quelle est la différence entre les requis (usage typique) pour ces deux types de serveurs (données versus métadonnées)? ii) En quoi ces requis mènent à un choix différent de configuration? **(1 point)**

i) Les métadonnées représentent un plus petit volume d'information. Cependant, leur cohérence est cruciale puisqu'elles sont la clé qui permet d'accéder aux données. A l'inverse, les données elles-mêmes peuvent être en très grand volume et arriver avec un très grand débit. ii) Ainsi, une configuration qui maximise le nombre de serveurs actifs en parallèle est avantageuse pour les données. Pour les métadonnées, une configuration active-passive, avec moins de serveurs actifs en parallèle, est suffisante et permet plus facilement d'en assurer la cohérence.

Le professeur: Michel Dagenais