POLYTECHNIQUE MONTRÉAL

Département de génie informatique et génie logiciel

Cours INF8480: Systèmes répartis et infonuagique (Automne 2021)
3 crédits (3-1.5-4.5)

CORRIGÉ DU CONTRÔLE PÉRIODIQUE

DATE: Lundi le 1er novembre 2021

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) Un service de fichiers prend 10ms 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 80 / seconde. Calculez N, le nombre moyen de requêtes dans le système, et R le temps de réponse moyen pour ce cas? Un second disque est ajouté au serveur qui peut maintenant traiter une nouvelle requête à chaque 5ms, mais le rythme des requêtes est aussi doublé à 160 / seconde. Que deviennent N et R? **(2 points)**

Le taux de service l dans le premier cas est de 1000ms/s / 10ms/r = 100r/s et le taux d'utilisation 80r/s / 100r/s = 0.8. En conséquence, N = 0.8 / (1 - 0.8) = 4, et R = N/l = 4 / 80/s = .05s ou 50ms. Dans le second cas, l est le même, 160r/s / 200r/s = 0.8 et N est donc le même à 4. Par contre, R = 4 / 160/s = 0.025 ou 25ms.

b) Un serveur doit être éteint pour en faire l'entretien. Il exécute une machine virtuelle qui doit donc être migrée vers un serveur de relève. L'image de cette machine virtuelle contient 12 000 000 pages et le lien réseau entre les deux serveurs permet de transférer 20 000 pages par seconde. La migration se fait initialement sans interrompre l'exécution de la machine virtuelle, copiant toutes les pages au premier tour et copiant aux tours subséquents les pages modifiées depuis le début du tour précédent. Cette machine virtuelle modifie 2000 pages par seconde. Lorsqu'il reste peu de pages modifiées, l'exécution de la machine virtuelle est arrêtée, les dernières pages modifiées sont copiées et l'exécution redémarre sur le serveur de relève. L'arrêt, pour copier les dernières pages modifiées, ne doit pas durer plus de 15ms. Combien de temps durera la migration au total? Combien de temps est-ce que l'exécution sera arrêtée? **(2 points)**

Le critère d'arrêt est de ne requérir que .015s pour copier les dernières pages, soit à 20 000 pages/s: .015s x 20 000 pages/s = 300 pages. Au premier tour, il faut copier les 12 000 000 pages en 12 000 000 pages / 20 000 pages/s = 600s. Pendant ce temps, 600s x 2000 p/s = 1 200 000 pages seront modifiées. Au second tour, il faut 1 200 000 pages / 20 000 pages/s = 60s pendant lesquelles 120 000 pages seront modifiées. Au troisième tour, il faut 120 000 pages / 20 000 pages/s = 6s pendant lesquelles 6s x 2000 p/s = 12 000 pages seront modifiées. Au quatrième tour, il faut 12 000 pages / 20 000 pages/s = 0.6s pendant lesquelles .6s x 2000 p/s = 1200 pages seront modifiées. Au cinquième tour, il faut 1200 pages / 20 000 pages/s = 0.06s pendant lesquelles .06s x 2000 p/s = 120 pages seront modifiées. On peut alors arrêter l'exécution, transférer ces pages en 120 pages / 20 000 pages / s = 0.006s. L'arrêt n'est donc que de 0.006s mais la migration totale a duré 600s + 60s + 6s + 0.6s + 0.06s + 0.006s = 666.666s.

c) Un nombre croissant de serveurs Web livrent des pages HTML qui contiennent du code Javascript. Expliquez en quoi le code Javascript ainsi envoyé peut aider la performance perçue par l'utilisateur, et diminuer la charge sur le serveur, par rapport à une page Web équivalente en HTML sans code exécutable comme le Javascript. **(1 point)**

Si la page est un formulaire, l'édition, la validation des champs et plusieurs fonctions semblables peuvent s'exécuter directement dans le fureteur Web du client. Cela peut aider la performance perçue par l'utilisateur en évitant la latence associée à faire plusieurs requêtes vers le serveur pour effectuer ces opérations d'édition et de validation. En réduisant le nombre de requêtes au serveur, on peut diminuer substantiellement la charge qui lui est imposée.

Question 2 (5 points)

a) Un groupe de 75 noeuds, sur un même réseau local, communique à l'aide de messages de groupe. Combien de messages au total seront envoyés sur le réseau pour envoyer un message de groupe selon un ordonnancement total, si la multi-diffusion est utilisée lorsqu'il y a plusieurs destinataires pour un même message? Si la multi-diffusion n'est pas disponible? Devrait-on ajouter des accusés de réception pour les messages de groupe avec ordonnancement total si i) de nouvaux messages arrivent très régulièrement et un certain délai pour détecter les messages perdus est tout à fait acceptable, ii) s'il est important de réenvoyer tout message perdu très rapidement et il peut y avoir de grandes variations dans les délais entre les messages de groupe? Expliquez. **(2 points)**

Il faut premièrement envoyer 1 message au serveur central d'ordonnancement total qui nous fournit le prochain numéro de séquence par 1 message de retour. Ensuite, on peut envoyer un message au 74 autres noeuds par multi-diffusion en 1 message, ou en 74 messages sans la multi-diffusion. Le total est donc de 3 avec la multi-diffusion, et 76 sans la multi-diffusion.

Si un message est perdu, le récipiendaire s'en rend compte rapidement lorsque le message suivant est reçu, puisqu'il y a un trou dans la séquence de la numérotation. Dans le cas i) on s'en rend compte d'autant plus rapidement que les messages sont fréquents. En outre, on spécifie que le délai est facilement toléré. Il n'est donc pas utile de demander des accusés de réception. Dans le second cas ii) les messages sont de fréquence variable et un long délai peut survenir avant le prochain message, alors qu'un tel délai n'est pas acceptable pour découvrir qu'un message manque et le retransmettre. Il faudrait donc avoir recours à des accusés de réception.

b) Un processus client effectue des requêtes auprès de nombreux services. Chaque requête demande l'envoi d'un message en 100us de CPU, le délai sur le réseau de 200us, le traitement sur le serveur de 900us (attente puis prise en charge par un des nombreux threads en parallèle), le délai sur le réseau de 200us pour la réponse et la réception de la réponse sur le client en 100us de CPU. Vous utilisez présentement un protocole de type appel de procédure à distance (RPC) synchrone. Quel est le nombre maximal de requêtes par seconde que peut effectuer le processus client dans ce cas avec un seul thread d'exécution? On vous propose de passer à un système basé sur des requêtes asynchrones (Messages Oriented Middleware). Un processus client peut alors envoyer plusieurs requêtes en rafale vers différents services et recevoir les réponses lorsqu'elles sont prêtes. Que devient alors le nombre maximal de requêtes que peut effectuer le client, toujours sur un seul thread? Montrez-le à l'aide d'un exemple où plusieurs requêtes asynchrones sont envoyées. **(2 points)**

Dans le premier cas, tout est effectué séquentiellement et une requête prend 100 + 200 + 900 + 200 + 100 = 1500us ou 1000000us/s / 1500us/r = 666.66r/s. Dans le second cas, le client peut envoyer 14 requêtes de suite en 1400us, avant de recevoir ensuite sans attendre les 14 réponses qui sont alors prêtes en 1400us. Cela fait donc 14 requêtes en 2800us soit 200us par requête ou 1000000us/s / 200us/r = 5000r/s. En effet, dans le deuxième cas, la latence de réponse des services est cachée par la profondeur de la queue d'envoi des requêtes asynchrones.

c) Quels sont les différents services de stockage disponibles sur un système infonuagique comme AWS ou OpenStack? Décrivez brièvement leurs caractéristiques. **(1 point)**

Le stockage d'instance (instance storage) est initialisé avec l'image de la machine virtuelle, est propre à l'instance, et n'est pas sauvé entre deux exécutions de cette image de machine virtuelle. Le stockage de blocs (e.g. Elastic block store) est permanent mais ne peut être attaché qu'à une seule instance à la fois. Le stockage d'objets (e.g. S3 Simple Storage Service) est permanent aussi et peut être accédé de plusieurs instances à la fois. Toutefois, il est seulement possible de remplacer au complet le contenu, on ne peut pas réécrire seulement une partie d'un "objet de stockage" (fichier). Le stockage de fichiers partagés (e.g. EFS Elastic File System) est une arborescence de fichiers permanente qui peut être accédée de plusieurs instances à la fois.

Question 3 (5 points)

a) Un processus serveur reçoit des requêtes de clients pour consulter l'inventaire des marchandises disponibles et faire des achats. Lorsqu'une requête est initiée, un objet Java RMI est créé dans le serveur et retourné au client pour la durée de son magasinage. Lorsque le client a terminé ses achats (complété sa transaction ou quitté son panier sans acheter), le proxy de l'objet Java RMI est libéré, et le processus serveur est averti pour libérer à son tour l'objet Java RMI créé pour ce client. On calcule qu'un nouveau client commence son magasinage à chaque 5 secondes. Par ailleurs, on mesure qu'il y a en tout temps dans le processus serveur 65 objets Java RMI exportés à des clients pour leur magasinage. Quelle est la durée moyenne de la session de magasinage d'un client? Pourquoi se fie-t-on sur un message du client pour savoir quand un objet n'est plus utilisé et peut être libéré, plutôt que d'utilliser l'algorithme de ramasse-miettes (garbage collector) déjà utilisé en Java pour les objets conventionnels? **(2 points)**

Puisque les requêtes arrivent au taux de 1 par 5 secondes, soit 0.2 r/s, et qu'il y a 65 requêtes actives en même temps, cela veut dire que chaque session de magasinage dure R = N/l soit 65r / 0.2r/s = 325s. Un ramasse-miettes fonctionne mal en réparti, puisqu'il faudrait arrêter tous les processus impliqués, le temps de noter quels sont les objets utilisés (rejoignables par des pointeurs locaux ou des références réseau). A la place, un système plus simple comme le décompte de référence est utilisé. Il se base soit sur des baux donnés aux utilisateurs externes des objets exportés, soit sur des messages de notification demandés aux utilisateurs externes des objets exportés.

b) Docker est un système très populaire pour les conteneurs. Quel est approximativement le surcoût de rouler une application dans Docker, plutôt que nativement sur le système d'exploitation? Si une application Linux a été encapsulée dans une image Docker, est-ce que cette image peut facilement être exécutée avec l'environnement Docker sous Windows? Si une application Linux x86 64 bits a été encapsulée dans une image Docker, est-ce que cette image peut facilement être exécutée avec l'environnement Docker sur un ordinateur ARM 64 bits? **(2 points)**

Docker ajoute une couche très fine pour intercepter les appels système, les transmettre au système d'exploitation, et rediriger les numéros de ports TCP/IP. Le surcoût par rapport à exécuter l'application nativement est en général très faible. L'environnement d'exécution de Docker permet d'intercepter les appels systèmes Linux pour les envoyer à des fonctions équivalentes sous Windows. Il est donc possible de rouler les images Docker, initialement prévues pour Linux, sous Windows. Docker ne fait pas de traduction ou d'émulation de répertoires d'instructions, on

ne peut rouler une image Docker x86 sous ARM directement. Docker peut toutefois faire appel à un émulateur comme QEMU, par exemple pour rouler une image ARM sous x86.

c) Le système de fichiers réparti GlusterFS permet des volumes avec le partage de charge par fichiers (striped volume). Expliquez en quoi cela consiste, comment cela apparaît dans le répertoire de chaque serveur GlusterFS, et ce que cela permet d'obtenir. **(1 point)**

Ce type de volume répartit le contenu de gros fichiers sur plusieurs briques (disques). Le fichier apparaîtra dans le répertoire de chaque serveur GlusterFS, mais son contenu sera différent sur chacun. En effet, le fichier contiendra la fraction des données destinée au serveur correspondant, et le reste du fichier contiendra des valeurs nulles en utilisant des trous dans le fichier (blocs non alloués et donc implicitement à 0). L'intérêt d'un tel type de volume est pour stocker de très gros fichiers, plus gros qu'un seul disque, et pour pouvoir faire la lecture ou l'écriture de morceaux en parallèle.

Question 4 (5 points)

a) Un volume d'un service de fichiers GlusterFS est répliqué sur 4 serveurs (i.e. chaque fichier se retrouve en 4 copies). Chaque serveur possède 7 disques. Chaque disque peut effectuer 120 accès (lecture ou écriture) par seconde. Les clients font des requêtes, accès en lecture ou en écriture, au serveur. Quel est le nombre maximal de lectures (s'il n'y a que des lectures) par seconde que pourrait soutenir ce service répliqué sur 4 serveurs, en supposant que la charge est répartie uniformément sur les serveurs et les disques, et que les disques constituent le facteur limitant? Quel est le nombre maximal d'écritures (s'il n'y a que des écritures) par seconde que pourrait soutenir ce service? **(2 points)**

En lecture, les serveurs opèrent en parallèle. Avec 4 serveurs de 7 disques à 120 accès/s, on a une capacité maximale de $4 \times 7 \times 120 = 3360$ *lectures par seconde. En écriture, chaque écriture doit être faite sur chacun des 4 serveurs. On peut donc soutenir 4 fois moins d'écritures que de lectures, soit 840 écritures par seconde.*

b) Un serveur NFS est accédé par de nombreux clients. Chaque client demande en moyenne 5 lectures de bloc et 1 écriture de bloc par seconde, dans l'arbre de répertoires accédé par NFS. Cependant, 90% des blocs accédés en lecture sont dans le cache du client NFS. De ceux-ci, 25% ont été validés récemment (moins de 3 secondes), alors que le 75% restant ne l'a pas été. Lorsqu'une demande de validation est envoyée au seveur, la réponse est positive dans 70% des cas, alors que dans 30% des cas la réponse est négative et le bloc en cache client ne peut donc pas être utilisé. Quel est le nombre de demandes de validation, de lectures de bloc et d'écritures de bloc par seconde que recevra le serveur si 200 clients y sont connectés? **(2 points)**

Les écritures sont toujours faites directement sur le serveur. Celui-ci verra donc 200 clients x 1 écriture / client-seconde = 200 écritures / seconde.

Les validations demandées représentent le cas où le bloc est en cache mais non validé récemment, soit 200 clients x 5 lectures / client-seconde x 0.90 x 0.75 = 675 validations / seconde. Les lectures sur le serveur se produisent dans deux cas, lorsque le bloc n'est pas en cache, 200 clients x 5 lectures / client-seconde x 0.10 = 100 lectures / seconde, ou lorsque le bloc est en cache mais

n'est plus valide, 200 clients x 5 lectures / client-seconde x 0.90 x 0.75 x 0.30 = 202.5 lectures / seconde, pour un total de 302.5 lectures / seconde.

c) Dans le cadre du travail pratique 2, un fichier nommé `operation.proto` devait être complété. A quoi sert ce fichier? Quel genre d'information contient-il? Quel programme lit ce fichier en entrée dans le `Makefile` et quelle est la sortie dérivée de ce fichier? **(1 point)**

Le fichier `operation.proto` est un fichier de définition d'interface (IDL) pour le protocole protobuf. Il sert à déclarer les fonctions des services à déployer (le message reçu en entrée et celui retourné en sortie) ainsi que le type des messages échangés (nom et type des champs de ces messages). Ce fichier est lu par le compilateur d'interface `protoc` et génère un fichier de code source en sortie.

Le professeur: Michel Dagenais