POLYTECHNIQUE MONTRÉAL

Département de génie informatique et génie logiciel

Cours INF8480: Systèmes répartis et infonuagique (Automne 2023)
3 crédits (3-1.5-4.5)

CORRIGÉ DU CONTRÔLE PÉRIODIQUE

DATE: Lundi le 30 octobre 2023

HEURE: 12h45 à 14h35

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 module de micro-service dans un système réparti reçoit de nombreuses requêtes. Chaque requête occupe un coeur de CPU pendant 10ms et en plus, dans 20% des cas, un disque pendant 100ms. Ce service est exécuté sur 8 coeurs de CPU et 16 disques qui lui sont dédiés. On suppose que les requêtes sont bien réparties entre les coeurs de CPU et entre les disques. Chaque client qui fait appel à ce service génère en moyenne 10 requêtes par seconde. 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 bien utiliser toutes les ressources disponibles? **(2 points)**

Puisque chaque requête prend 10ms sur un coeur de CPU, chaque coeur peut servir 1000ms/s / 10ms/r = 100 requêtes / seconde. Le total est donc de 800 requêtes / seconde pour les 8 coeurs de CPU. Chaque requête prend en moyenne 0.2 x 100ms = 20ms de disque. Chaque disque peut servir 1000ms/s / 20ms/r = 50 requêtes / seconde. Le total est donc de 800 requêtes / seconde pour les 16 disques. i) C'est donc un système très équilibré avec les disques et les coeurs de CPU à un niveau équivalent, les deux étant limitants au même niveau. ii) Le système sature donc avec 800 requêtes / seconde, ce qui est équivalent à (800 requête / seconde) / (10 requête / client-seconde) = 80 clients. Si chaque requête reste dans le système 10ms CPU + 20ms disque = 30ms en moyenne, cela demande 800 r/s x 1s/1000ms x 30ms-thread/r = 24 thread, soit 24 fils d'exécution.

b) Un serveur dans un commerce reçoit des requêtes qui 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 150 / seconde et le serveur peut traiter chaque requête en 5ms. i) Calculez N, le nombre moyen de requêtes dans le système, et W le temps de réponse moyen en secondes pour ce cas? On prévoit ouvrir une nouvelle succursale, avec un nouveau serveur qui recevra le même nombre de requêtes et aura la même capacité de traitement (i.e. deux fois plus de requêtes mais deux serveurs pour les traiter). ii) Que sera le temps d'attente moyen si chaque serveur a sa propre queue d'attente? iii) Que sera le temps d'attente moyen si une queue unique alimente les deux serveurs? **(2 points)**

La capacité de traitement u dans le premier cas est de 1000ms/s / 5ms/r = 200r/s, alors que le taux d'arrivée l est de 150r/s. Le taux d'utilisation est ainsi de l/u = 150r/s / 200r/s = 0.75. i) En conséquence, N = 0.75 / (1 - 0.75) = 3, et le temps d'attente moyen W = N/l = 3 / 150r/s = 0.02s ou 20ms. ii) Dans le second cas, si on a 2 succursales identiques, avec chacune le même taux d'arrivée et de service que dans le premier cas, rien ne change et W restera 0.02s ou 20ms. iii) Cependant, si on a une queue d'attente unique, la situation est différente. Le taux d'arrivée et le taux de traitement global double. La taux d'utilisation reste le même, 300r/s / 400r/s = 0.75. En première approximation, on peut estimer le nombre moyen de requêtes dans le système en supposant un seul serveur avec le double de la capacité, N = 0.75 / (1 - 0.75) = 3. Le temps moyen d'attente diminuerait donc à W = 3 / 300r/s = 0.01s ou 10ms. Si on prend une formule qui tient spécifiquement compte du fait qu'on a deux serveurs en parallèle, pour le temps d'attente en queue Wq, on obtient $Wq = l^2/(u(4u^2 - l^2)) = 0.00642s$ *et donc le temps total est de* $6.42ms + 5ms = 11.42ms$*, ce qui est assez près de notre estimé.*

Il est normal que le temps d'attente diminue avec une queue unique, on enlève l'inefficacité associée à 2 queues lorsqu'une est pleine alors que l'autre est vide et son unité de traitement est sous-utilisée.

c) Dans les systèmes client-serveur, on permet parfois aux clients d'envoyer du code à exécuter sur le serveur. Un exemple de cela est les procédure envoyées par les clients à exécuter sur un serveur de base de données. Quels sont les problèmes associés à une telle manière de faire? Quels sont les avantages possibles de permettre l'exécution de telles procédures sur le serveur? **(1 point)**

En permettant aux clients d'envoyer des procédures qui s'exécutent sur le serveur, il y a un risque de surcharger le serveur et cela cause plus de risques au niveau de la sécurité du serveur. Cela peut être utile si le client n'a pas la capacité de traitement nécessaire pour faire le calcul rapidement. Un avantage important est que cela peut sauver beaucoup de transferts de données coûteux, si la procédure requiert l'accès à beaucoup de données qui se trouvent sur le serveur.

Question 2 (5 points)

a) Un groupe de 60 processus, sur autant de noeuds connectés sur le même réseau local, communiquent à l'aide de messages de groupe. Un processus veut envoyer un message fiable aux autres processus. i) Comment cela peut-il être réalisé? ii) Combien de messages seront envoyés sur le réseau par les différents processus, si la multi-diffusion est disponible? iii) Si la multi-diffusion n'est pas disponible? iv) Pour un message de groupe totalement ordonnancé, avec multi-diffusion, que deviendra le nombre de messages envoyés? On ne compte pas les retransmissions causées par les messages perdus. **(2 points)**

Pour avoir un envoi de message fiable, il faut recevoir un accusé de réception de chaque récipiendaire. Ainsi, le processus enverra un message en multi-diffusion aux 59 autres membres. i) Il recevra ensuite un accusé de réception de chacun, pour un total de 60 messages. ii) Si la multi-diffusion n'est pas disponible, le processus enverra 59 messages et recevra 59 accusés de réception, pour un total de 118 messages. iii) Pour un envoi totalement ordonnancé, on a un message pour obtenir un numéro d'ordonnancement, une réponse, et un envoi en multi-diffusion du message avec son numéro d'ordre, pour un total de 3 messages. L'accusé de réception n'est pas nécessaire car un processus découvrira vite si un message lui manque, en voyant le bris dans les numéros d'ordonnancement.

b) Un appel de procédure à distance doit contenir les arguments suivants: string prenom, string nom, int question, int note. Si les valeurs sont: "Jean-Claude", "Bouchard-Tremblay", "10", "350", combien d'octets seront requis pour encoder cette information avec CORBA CDR (32 bits)? 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 11 lettres de "Jean-Claude", 4 octets pour la longueur du nom et 20 octets (multiple de 4) pour les 17 lettres de "Bouchard-Tremblay", 4 octets pour la longueur de question et 4 octets pour la longueur de note. Le total est donc de 4+12+4+20+4+4 = 48 octets. Pour le format protobuf, nous avons pour le prenom 1 octet type/champ, 1 octet longueur et 11 octets pour

"Jean-Claude", pour le nom 1 octet type/champ, 1 octet longueur et 17 octets pour "Bouchard-Tremblay", pour le numéro de question 1 octet type/champ et 1 octet pour la valeur 10, et pour la note 1 octet type/champ et 2 octets pour 200. Le total est de 1+1+11+1+1+17+1+1+1+2 = 37 octets.

c) Ces dernières années, les intergiciels orientés message (MOM), comme ZeroMQ, sont devenus très populaires et ont en grande partie remplacé les intergiciels d'appel de procédure à distance (RPC). Par exemple, OpenStack utilise RabbitMQ et ZeroMQ. Quels sont les avantages d'utiliser un intergiciel orienté message plutôt qu'un intergiciel d'appel de procédure à distance? **(1 point)**

Les intergiciels orientés message sont plus complets et flexibles que les systèmes d'appel de procédure à distance. En effet, ils peuvent avoir des communications un à un synchrones requête-réponse, comme les appels de procédures à distance. Cependant, ils permettent de faire beaucoup plus avec des modes comme publication-souscription et bien d'autres et permettent les communications autant synchrones qu'asynchrones.

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 *client_request_handler* est créé et une référence réseau à cet objet est retournée au client de la requête. Le client cesse d'utiliser cette référence réseau en moyenne 240 secondes plus tard, en envoyant un message pour se désenregistrer de cet objet réseau exporté. Le processus serveur peut alors libérer l'objet de type *client_request_handler* en question. i) Si le processus serveur reçoit 1 requête par 5 secondes de chaque client, et sert 400 clients, combien d'objets de type *client_request_handler* sont présents simultanément dans le processus serveur en moyenne? ii) Dans un système d'appel de méthode à distance comme Java RMI, quel est le contenu de la table des objets exportés? Quel est le contenu de la table des objets importés? **(2 points)**

Le temps moyen qu'un objet reste alloué pour une requête est de 240s. Le taux d'arrivée des requêtes est de 1r/5s par client x 400 clients = 80 r/s. i) Le nombre total d'objets alloués simultanément est donc de 240s x 80 r/s = 19200. ii) La table des objets exportés contient pour chaque objet exporté un pointeur vers l'objet en question, son identifiant réseau et une liste des processus distants qui l'utilisent. La table des objets importés contient pour chaque objet importé son identifiant réseau (avec le processus d'où il provient) et un pointeur vers le proxy créé localement pour représenter cet objet distant.

b) Une machine virtuelle tourne sur un noeud physique A. L'image de cette machine virtuelle contient 4 000 000 pages. On veut migrer cette machine virtuelle vers un noeud B à travers un lien réseau qui permet d'envoyer 10 000 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 2000 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 0.2 seconde. Combien de temps durera la migration au total? Combien de temps est-ce que l'exécution sera arrêtée? **(2 points)**

Lors du premier tour, les 4 000 000 pages seront copiées en 4 000 000 / 10 000 = 400s. Pendant ces 400s, nous aurons 400 x 2000 = 800 000 pages modifiées. Au second tour, ces pages sont copiées en 800 000 / 10 000 = 80s, temps pendant lequel 80 x 2000 = 160 000 pages sont modifiées. Au troisième tour, les pages sont copiées en 160 000 / 10 000 = 16s, et 16 x 2000 = 32 000 pages sont modifiées. Au quatrième tour, les pages sont copiées en 32 000 / 10 000 = 3.2s, et 3.2 x 2000 = 6400 pages sont modifiées. Au cinquième tour, les pages sont copiées en 6400 / 10 000 = 0.64s, et 0.64 x 2000 = 1280 pages sont modifiées. Rendu là, l'exécution est interrompue, les pages restantes prenant moins que 0.2s à copier, et les 1280 pages restantes sont copiées en 1280 / 10 000 = 0.128s. Le temps total pour la migration est donc de 400s + 80s + 16s + 3.2s + 0.64s + 0.128s = 499.968s. L'exécution est arrêtée pendant le dernier tour qui dure 0.128s.

c) Quels sont les avantages et inconvénients d'utiliser des conteneurs par rapport à des machines virtuelles? Donnez un exemple d'application où chacun serait plus approprié. **(1 point)**

Les machines virtuelles permettent toute latitude quant à la configuration utilisée, incluant de choisir le système d'exploitation. C'est aussi une solution qui permet a priori une meilleure isolation au niveau de la sécurité informatique. Par contre, les conteneurs ont le grand avantage de présenter un surcoût très bas, par rapport à s'exécuter directement sur le système d'exploitation d'un noeud physique, sans couche d'isolation. Cet avantage en performance est particulièrement important au niveau du temps de démarrage. Le démarrage d'un conteneur prend typiquement une fraction de seconde, comparé à une machine virtuelle qui prendra de l'ordre d'une à plusieurs minutes. Si on veut offrir un service à des clients où chacun peut installer le système d'exploitation de son choix, les machines virtuelles seront le choix évident. Si on veut déployer un service, par exemple une application de commerce en ligne, où on peut rapidement ajuster le nombre d'instances parallèles, les conteneurs sont un choix tout indiqué.

Question 4 (5 points)

a) Un serveur NFS sert de nombreux clients. Les processus sur chaque client effectuent en moyenne 2 écritures et 8 lectures par seconde sur des blocs de fichiers venant de ce serveur. Les blocs accédés en lecture se trouvent en cache sur le client dans 85% des cas. Parmi les blocs en cache, 60% ont été validés depuis moins de 3 secondes. Les autres blocs en cache demandent une validation auprès du serveur. Parmi ces blocs qui demandent une validation, 40% ont été modifiés et nécessitent une lecture sur le serveur en plus, alors que 60% sont valides. L'écriture d'un bloc sur le serveur prend 20ms de disque. La lecture d'un bloc du serveur prend 15ms de disque dans 30% des cas, et est servie à partir du cache d'entrée-sortie en temps négligeable dans 70% des cas. Une validation d'un bloc du serveur prend 15ms de disque dans 10% des cas, et est servie à partir du cache d'entrée-sortie en temps négligeable dans 90% des cas. Quel est le nombre de clients maximal que peut soutenir le serveur sans être saturé, s'il contient 16 disques, que les coeurs de CPU ne sont pas un facteur significatif, et que les requêtes sont réparties uniformément entre les disques? **(2 points)**

Lorsqu'un processus sur un client effectue une écriture sur un bloc venant d'un serveur, ceci se traduit automatiquement en une écriture sur le serveur. Lorsqu'un processus sur un client effectue une lecture d'un bloc venant d'un serveur, cela se décompose comme suit:

- *En cache et validé depuis moins de 3s: 0.85 x 0.6 = 0.51, rien à faire*
- *En cache, demande validation, validation positive: 0.85 x 0.4 x 0.6 = 0.204, demande de validation*
- *En cache, demande validation, validation négative: 0.85 x 0.4 x 0.4 = 0.136, demande validation et lecture*
- *Pas en cache, 0.15, demande lecture*
- *Total: 0.204 + 0.136 = 0.34 validation, 0.136 + 0.15 = 0.286 lecture*

Chaque client, à chaque seconde, demande donc au serveur 2 écritures, 8 x 0.34 = 2.72 validations, et 8 x 0.286 = 2.288 lectures. La lecture d'un bloc sur le serveur prend 0.3 x 15ms = 4.5ms de disque en moyenne. Une validation prend 0.1 x 15ms = 1.5ms de disque en moyenne. Une écriture prend 20ms. Un client prend donc par seconde en moyenne 2 écritures x 20ms + 2.72 validations x 1.5ms + 2.288 lectures x 4.5ms = 54.376ms. Avec 16 disques, il peut donc soutenir jusqu'à 16 x 1000ms/s / 54.376ms/client = 294 clients.

b) Un volume sur un service GlusterFS est configuré pour être Réparti, Répliqué (Distributed Replicate). Les serveurs S1, S2 et S3 offrent chacun une brique de disque de même dimension qui sont ainsi répliquées et forment le volume répliqué VR0. Les serveurs s4, s5 et s6 font de même et forment le volume répliqué VR1. Finalement, le volume réparti répliqué VRR distribue ses fichiers entre VR0 et VR1. Si 8 fichiers, nommés `fic0` à `fic7`, sont placés sur ce volume et se distribuent équitablement entre les différentes locations possibles, donnez une répartition possible de ces 8 fichiers sur les 6 serveurs. Indiquez bien quels fichiers se retrouvent sur la brique de chaque serveur. On suppose que le seul facteur important pour la performance est l'accès aux disques. Si chaque serveur a un disque qui permet l'écriture ou la lecture de fichiers à 50MiO/s, quelle serait la bande passante pour écrire des fichiers sur ce volume VRR? Pour lire des fichiers? **(2 points)**

Les fichiers vont se répartir entre VR0 et VR1. Ensuite, chaque fichier sur VR0 ou VR1 sera répliqué sur les 3 serveurs qui les composent. Ceci pourrait donner: S1 contient (fic0, fic2, fic4, fic6), S2 (fic0, fic2, fic4, fic6), S3 (fic0, fic2, fic4, fic6), S4 (fic1, fic3, fic5, fic7), S5 (fic1, fic3, fic5, fic7) et S6 (fic1, fic3, fic5, fic7). Au moment de lire, chaque serveur peut être accédé en parallèle, ce qui donne une bande passante totale de 6 x 50MiO/s = 300MiO/s. Pour écrire, trois serveurs sont occupés à écrire la même chose. Nous avons donc seulement les deux volumes distribués qui sont en parallèle, pour une bande passante totale nette de 2 x 50MiO/s = 100MiO/s.

c) Quelle(s) technique(s) les systèmes de fichiers répartis comme GlusterFS et CEPH utilisent pour permettre une très grande mise à l'échelle? **(1 point)**

Lorsqu'on veut pouvoir servir un nombre de clients arbitrairement grand, il faut s'assurer de pouvoir ajouter des serveurs en conséquence, sans qu'il y ait de point central qui devienne un goulot d'étranglement. Il faut donc pouvoir répartir les clients entre les serveurs, sans besoin de passer par un répartiteur central. Une technique efficace pour ce faire est l'utilisation de

fonctions de hachage qui disent pour chaque fichier, ou morceau de fichier, quel est le serveur qui en est responsable. La réplication, en lecture, permet d'augmenter le nombre de serveurs qui peuvent offrir en parallèle un fichier populaire. Par contre, pour l'écriture, cela pose des problèmes pour maintenir la cohérence entre un grand nombre de copies.

Le professeur: Michel Dagenais