POLYTECHNIQUE MONTRÉAL

Département de génie informatique et génie logiciel

Cours INF8480: Systèmes répartis et infonuagique (Hiver 2020)
3 crédits (3-1.5-4.5)

CORRIGÉ DU CONTRÔLE PÉRIODIQUE

DATE: Vendredi le 28 février 2020

HEURE: 9h30 à 11h20

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 client envoie une requête de 4000 octets à un serveur et reçoit une réponse de 10 000 octets. Le client prend 200us pour préparer sa requête, le serveur 800us pour y répondre, et le client 100us pour traiter la réponse. La transmission de la requête et de la réponse se font sur un réseau qui a une latence de 100us et un débit de 10 000 000 octets / seconde. Combien de temps au total prend une requête? Combien de requêtes par seconde le client peut-il donc faire à ce serveur avec un seul thread d'exécution? Le client et le serveur ne font rien d'autre et ne sont donc pas interrompus par d'autres tâches. **(2 points)**

Le débit de 10 000 000 o/s représente 10 o/us. L'envoi sur le réseau prend 100 us + 4000 o / 10 o/us = 500 us. La réponse prendra 100 us + 10000 o / 10 o/us = 1100 us. Le total avec le traitement est donc 200 us + 500 us + 800 us + 1100 us + 100 us = 2700 us. On peut donc servir 1 000 000 us/s / 2700 us/r = 370 requêtes par seconde.

b) Une machine virtuelle tourne sur un noeud physique A. L'image de cette machine virtuelle contient 8 000 000 pages. On veut migrer cette machine virtuelle vers un noeud B 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 1000 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 0.25 seconde. 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 .25s pour copier les dernières pages, soit à 25 000 pages/s: .25s x 25 000 pages/s = 6250 pages. Au premier tour, il faut 8 000 000 pages / 25 000 pages/s = 320s. Pendant ce temps, 320s x 1000 p/s = 320 000 pages seront modifiées. Au second tour, il faut 320 000 pages / 25 000 pages/s = 12.8s pendant lesquelles 12 800 pages seront modifiées. Au troisième tour, il faut 12 800 pages / 25 000 pages/s = .512s pendant lesquelles 512 pages seront modifiées. On peut alors arrêter l'exécution, transférer ces pages en 512 pages / 25 000 pages / s = .02s. L'arrêt n'est donc que de 0.02s mais la migration totale a duré 320s + 12.8s + .512s + .02s = 333.3s.

c) Un serveur prend 100us pour traiter chaque requête. Le serveur utilise un seul thread, sur un coeur dédié, pour traiter les requêtes, et les requêtes sont mises en queue d'attente lorsque le serveur est déjà occupé avec une requête. Si le serveur reçoit en moyenne 9500 requêtes par seconde, quel est le nombre moyen de requêtes en queue sur le serveur? **(1 point)**

Le serveur peut traiter 1 000 000 us/s / 100 us/r = 10000 r/s. Le taux d'utilisation du service est donc de 9500 / 10000 = 0.95. On peut en déduire que le nombre moyen de requêtes en queue est de $\overline{L} = 0.95^2/(1-0.95) = 18.05$*, alors que le nombre moyen de requêtes dans le système est de* $\overline{N} = 0.95/(1-0.95) = 19$.

Question 2 (5 points)

a) Sur un réseau qui supporte la multi-diffusion, un message de groupe atomique est envoyé d'un membre aux autres. Le groupe compte N membres. Si sur ce réseau 1 message sur 4 envoyé par chaque noeud est perdu (i.e. pour cet échange, le premier message envoyé par un noeud donné est perdu, le second, troisième et quatrième passent, le cinquième est perdu...), combien de messages envoyés au total seront requis pour compléter ce message atomique. **(2 points)**

L'envoyeur envoie un message par multi-diffusion et attend les confirmations. Le message étant perdu, il ne reçoit rien et renvoie son message. Les N-1 accusés de réception des autres membres du groupe sont perdus. L'envoyeur envoie un troisième message, les N-1 accusés de réception parviennent alors à l'envoyeur. Il envoie en conséquence son message de confirmation. Le total est donc de 1 + 1 + (N - 1) + 1 + (N - 1) + 1 = 2N + 2.

b) Un message doit être envoyé avec gRPC utilisant l'encodage protobuf. Ce message suit la structure Recette et a pour contenu `{{"eau",80}, {"levure",5}, {"sucre",5}, {"farine",250}, {"sel",2}, {"yogourt",15}}`. Quel est le nombre d'octets requis pour encoder le message suivant sur un ordinateur 32 bits? 64 bits? **(2 points)**

```
message Ingredient {
  string nom = 1;
  int32 quantite = 2;
}

message Recette { repeated Ingredient = 1; }
```

Les ingrédients prennent 1 octet pour le champ/type "nom", 1 octet pour la longueur du "nom" plus 1 octet par caractère dans le nom, 1 octet pour le champ/type "quantite" et 1 octet pour la quantité, sauf pour la "farine" qui prend 1 octet supplémentaire puisque 250 n'entre pas sur 7 bits. Le total pour les 6 ingrédients est donc de 6 champs x 4 octets + 1 octet (pour la continuation de 250, quantité de farine) = 25 octets, plus le nombre total d'octets dans les noms 3 + 6 + 5 + 6 + 3 + 7 = 30, pour un total de 25 + 30 = 55 octets. Pour la structure Recette qui comprend 6 fois le champ répété Ingredient, selon la version de protobuf, on pourrait avoir 1 octet pour le champ/type de chaque Ingredient dans Recette, ce qui ajoute 6 octets, pour un total de 55 + 6 = 61 octets. Le fait d'avoir un ordinateur 32 ou 64 bits ne change rien avec les varints.

c) Quelle est la différence entre la sémantique "peut-être" et la sémantique "au plus une fois" pour les appels de procédure à distance? **(1 point)**

Pour la sémantique peut-être, le message n'est envoyé qu'une seule fois sans demander d'accusé de réception. Il peut ou non se rendre. Rarement, le réseau pourrait même répéter le paquet. La sémantique au plus une fois renvoie la requête jusqu'à obtention d'une réponse. Pour ne pas que le service puisse voir plus d'une fois la même requête, lors d'une seconde réception de la même requête, on suppose que la réponse envoyée n'a pas été reçue et on renvoie la réponse qui avait été mémorisée, sans traiter la requête une seconde fois. Cette sémantique est en fait "exactement une fois", puisqu'on renvoie la requête jusqu'à obtention d'une réponse, sauf si le service est perpétuellement en panne.

Question 3 (5 points)

a) Trois serveurs CODA redondants sont pris pour servir des clients. Tous les fichiers lus ou écrits par les clients se retrouvent déjà dans leur cache CODA, et n'ont pas besoin d'être revalidés car le client se fie sur le service de notification des serveurs. Des étudiants sont en train de travailler sur un intéressant projet de programmation. Ils modifient et sauvent le code source, de 64KiO, et ensuite créent une nouvelle version de l'exécutable compilé, de 128KiO. Ceci se répète sur chaque client à chaque 5 minutes, le temps de trouver et corriger le prochain bogue. A chaque sauvegarde du fichier source, ou création du fichier exécutable par le compilateur, l'application ferme le fichier et son contenu est propagé en séquence, bloc par bloc, aux trois serveurs redondants. Le serveur peut lire ou écrire un bloc de 8KiO en 2ms de coeur de CPU et 8ms de disque. Combien de clients ces trois serveurs redondants peuvent-ils supporter, s'ils contiennent chacun 32 coeurs de CPU et 16 disques? **(2 points)**

Chaque client envoie 8 blocs de 8KiO pour le source et 16 blocs de 8KiO pour l'exécutable, à chaque 5 minutes, soit 24 blocs / (5m x 60s/m) = .08 bloc/s. Chaque serveur est occupé par 1 bloc pendant 2ms de CPU et 8 ms de disque. Il peut donc servir 32 x 1000 ms/s / 2 ms/r = 16000 requêtes de bloc par seconde avec son CPU et 16 x 1000 ms/s / 8 ms/b = 2000 b/s avec ses disques. Les disques seront donc le facteur limitant. Puisque chaque client demande .08 b/s, le serveur pourra satisfaire 2000 b/s / .08 b/s = 25000 clients. Puisque les 3 serveurs redondants font le même travail en même temps, cela n'ajoute pas de capacité par rapport au calcul fait pour un seul serveur.

b) Le nouvel épisode de Pollux à Polytechnique vient de sortir et est offert par BitTorrent. Une première copie, qui contient les 1000 blocs de 1MiO, est disponible sur un serveur de départ. Les 8000 étudiants de Polytechnique en veulent une copie le plus tôt possible dès sa sortie. Tous sont connectés sur un réseau très rapide qui n'est pas un facteur limitant. Le serveur de départ et les ordinateurs des 8000 étudiants permettent tous de recevoir et de transmettre simultanément 1MiO par seconde dans chaque direction. Quel sera le temps requis avant que tous en obtiennent une copie? **(2 points)**

Pour que les 8000 étudiants aient une copie complète, il faudra transférer 8000 x 1000 blocs = 8 000 000 blocs de 1MiO. Pour la première seconde, 1 seul transfert est possible (serveur de départ), pour la deuxième seconde 2, ensuite 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8000 (on ne peut pas avoir plus de 8000 ordinateurs étudiants qui reçoivent des blocs en même temps). A chaque seconde suivante on peut encore avoir 8000 transferts au maximum. Le total pour les 13 premières secondes est de 8191 blocs. Ensuite, on peut transférer 8000 blocs à chaque seconde au maximum. Pour transférer les 8 000 000 blocs il faudra donc au minimum 13s + (8000000 b - 8191 b) / 8000 b/s = 1011.97s, soit 1012 secondes. On suppose ici qu'il est toujours possible pour un ordinateur étudiant de trouver un bloc qu'il n'a pas encore, ce qui devrait souvent être le cas avec BitTorrent qui favorise la propagation des blocs plus rares. Cela reste néanmoins une borne inférieure.

Une stratégie de diffusion systématique et relativement efficace est de commencer par envoyer un bloc différent à chacun de 1000 ordinateurs étudiants, ce qui prend 1000s. Ensuite, chacun des 1000 ordinateurs envoie son bloc à un autre ordinateur qui n'a encore rien, en 1s. Ces 2000 ordinateurs envoient ensuite chacun leur bloc à 2000 autres ordinateurs qui n'ont encore rien en

1s. Finalment, ces 4000 ordinateurs envoient chacun leur bloc aux 4000 autres ordinateurs en 1s. Ainsi, après 1003 secondes, chacun des 8000 ordinateurs étudiants a un bloc, et chaque bloc du fichier original se retrouve sur 8 ordinateurs étudiants différents. Dans les 999 secondes suivantes, chaque ordinateur, dans un groupe de 1000 qui contiennent collectivement les 1000 blocs du fichier, peut envoyer son bloc aux 999 autres ordinateurs du groupe. Après ces échanges, chaque ordinateur aura une copie complète du fichier, ce qui aura pris un total de 1000s + 3s + 999s = 2002s. On voit facilement que cette solution est une borne supérieure. En effet, pendant les premières 1000s, seul le serveur original et un seul client sont actifs, alors que ceux qui ont déjà reçu un bloc pourraient transmettre des blocs.

Une simulation naïve de clients bittorrent, selon le niveau de sophistication des heuristiques choisis pour décider quel client demande à quel autre un bloc, donne un temps total de 1150s (chaque client à tour de rôle est le premier à demander un bloc) ou de 1090 (le client avec le moins de blocs reçus demande en premier). Ceci est cohérent avec les bornes trouvées et s'approche passablement de la borne inférieure calculée.

c) Dans le travail pratique 3, plusieurs fichiers ont été créés avec la commande suivante. Donnez le contenu possible de /home/disk1, sur les conteneurs gluster1 et gluster2, pour les deux cas suivants de configuration: i) avec réplication, ii) avec distribution. **(1 point)**

```
for (( i =1; i <= 20; i++ ) ) ; do touch $i; done
```

Avec la réplication, les fichiers 1 à 20 se retrouveront chacun à la fois sur gluster1 et gluster2. Avec la distribution, les fichiers 1 à 20 se répartiront entre gluster1 et gluster2 selon la fonction de hachage utilisée par GlusterFS. Par exemple, on pourrait avoir les fichiers pairs sur gluster1 et impairs sur gluster2.

Question 4 (5 points)

a) Dans une entreprise, chaque client effectue en moyenne 1 requête à un serveur donné à chaque 5 secondes. Le serveur peut répondre à 85% des requêtes en 1ms de CPU (information en mémoire), à 10% des requêtes en 2ms de CPU et 10ms de disque (information sur disque), et à 5% des requêtes en accédant d'autres serveurs en 5ms de CPU, 10ms de disque et 200ms d'attente après les autres serveurs. Ce service utilise plusieurs threads et le serveur possède 16 coeurs de CPU et 10 disques. Quel est le nombre maximal de clients que ce serveur peut desservir efficacement? Quel est le nombre minimal de threads qu'il devrait utiliser? **(2 points)**

Chaque client demande par requête .85 x 1ms + .1 x 2ms + .05 x 5ms = 1.3ms de CPU, .1 x 10ms + .05 x 10ms = 1.5ms de disque et .05 x 200 = 10ms d'attente. Les 16 coeurs peuvent servir: 16 coeurs x 1000 ms/s / 1.3 ms/r = 12307 r/s alors que les 10 disques peuvent servir 10 x 1000 ms/s / 1.5 ms/r = 6666 r/s. Les disques sont le facteur limitant et à 1 requête par 5 seconde par client, le service peut accomoder 5 s/r x 6666 r/s = 33333 clients. Chaque requête reste en moyenne 1.3ms + 1.5ms + 10ms = 12.8ms dans le système, alors qu'une nouvelle requête est servie à chaque 1 / 6666 r/s (ou 1.5ms / 10 disques) = 0.15ms. Le nombre de requêtes simultanées dans le système est donc de 12.8ms / 0.15ms = 85.3, il faut donc au moins 86 threads.

b) Un système client serveur utilise des objets Java qui communiquent avec Java RMI. Au moment d'une nouvelle requête, un objet Java RMI est créé sur le serveur et est accédé par le client. Lorsque la requête est terminée, le proxy sur le client est libéré et un message est envoyé au serveur pour l'informer que l'objet Java RMI correspondant n'est plus utilisé et peut être libéré. Le serveur reçoit environ 100 requêtes par seconde et chaque requête dure environ 5 minutes. Combien d'objets Java RMI pour ce service existent simultanément dans le serveur en moyenne? Si dans 1% des cas, le client termine de manière catastrophique pendant sa requête et le message du proxy libéré n'est pas envoyé au serveur, qu'arrivera-t-il? Est-ce que le serveur pourra continuer sans problème, ou au bout de combien de temps, et selon quels paramètres, deviendra-t-il en difficulté? **(2 points)**

Un serveur qui démarre reçoit une requête. Cette requête sera terminée dans 5m x 60s/m = 300s. Entretemps, il recevra donc 300s x 100r/s = 30000 requêtes qui seront présentes simultanément dans le serveur. Par la suite, à chaque nouvelle requête qui entre, une sort et le nombre de requêtes simultanément dans le serveur reste environ le même. Si 1% des requêtes ne se terminent pas correctement et l'objet RMI n'est pas relâché, cela veut dire que 1 objet RMI par seconde s'accumule. Selon l'espace mémoire pris par l'objet RMI et la taille de mémoire disponible dans le serveur, cela prendra plus ou moins de temps avant que la mémoire ne se remplisse et que le serveur ne devienne très lent et se termine en erreur, faute de mémoire disponible.

c) Une entreprise veut offrir un service en utilisant des serveurs virtuels obtenus d'un fournisseur comme AWS ou Azure. Sur ces serveurs virtuels, elle peut installer Linux et OpenStack afin de déployer et d'orchestrer des machines virtuelles. A la place, elle peut installer Linux et Kubernetes afin de déployer et d'orchestrer des conteneurs. Pour chacune de ces deux solutions, dites si c'est faisable et si c'est efficace. **(1 point)**

Les serveurs "virtuels" obtenus du fournisseur représentent un premier niveau de virtualisation. Il est possible d'installer Linux et OpenStack, et de mettre des machines virtuelles dessus, mais cela veut dire des machines virtuelles imbriquées qui ne pourront bénéficier du support matériel pour la virtualisation, puisque le support matériel ne s'applique qu'au premier niveau. Le surcoût sera assez grand et ce n'est donc pas très efficace. Déployer des conteneurs sur des serveurs virtuels est à la fois faisable et efficace, puisqu'on reste à un seul niveau de virtualisation, qui peut alors être supporté par matériel.

Le professeur: Michel Dagenais