ECOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF8480: Systèmes répartis et infonuagique (Automne 2018)
3 crédits (3-1.5-4.5)

CORRIGÉ DE L'EXAMEN FINAL

DATE: Vendredi le 14 décembre 2018

HEURE: 13h30 à 16h00

DUREE: 2H30

NOTE: Aucune documentation permise sauf un aide-memoire, préparé par l'étudiant, qui consiste en une feuille de format lettre manuscrite recto verso, calculatrice non programmable permise

Ce questionnaire comprend 5 questions pour 20 points

Question 1 (4 points)

a) Un ordinateur contient un processeur à 8 coeurs, 4 disques de type SSD et 4 disques conventionnels. Il sert des requêtes DNS. Pour toutes les requêtes reçues, une recherche est effectuée en mémoire vive, ce qui prend 1ms de CPU. Ceci permet de servir 60% des requêtes. Pour les autres requêtes, dont la réponse ne se trouve pas en mémoire vive, une recherche est effectuée sur les disques SSD, ce qui prend 2ms de disque SSD. Ceci permet de servir la moitié des requêtes qui n'ont pas eu réponse en mémoire vive. Les requêtes restantes, dont la réponse n'a pas été trouvée en mémoire vive ni sur les disques SSD, seront servies à partir des disques conventionnels en 20ms de disque conventionnel. i) Quel est le nombre maximal de requêtes par seconde qui pourraient être servies si le logiciel traite séquentiellement les requêtes avec un seul thread? ii) Quel est le nombre maximal de requêtes servies si le logiciel utilise un grand nombre de threads et que les requêtes sont réparties uniformément sur les coeurs et les disques? Quel est le nombre minimal de thread requis pour maximiser le nombre de requêtes servies par seconde? **(2 points)**

Une requête demande en moyenne 1ms CPU, 0.4 x 2ms = 0.8ms de disque SSD et 0.5 x 0.4 x 20ms = 4ms de disque conventionnel, pour un total de i) 5.8ms ou 1000ms/s / 5.8ms/r = 172r/s. Avec de nombreux threads, les 8 coeurs pourraient traiter 8 x 1000ms/s / 1ms/r = 8000r/s, les disques SSD 4 x 1000ms/s / 0.8ms/r = 5000r/s, et les disques conventionnels 4 x 1000ms/s / 4ms/r = 1000r/s. Ce sont donc les disques conventionnels qui sont le facteur limitant et le nombre maximal de requêtes qui peuvent être servies est de ii) 1000r/s. Puisque chaque requête dure en moyenne 5.8ms sur un thread, cela donne 5.8ms-t/r x 1000r/s = 5800ms-t/s ou 5.8s de thread par seconde. Il faut donc au moins 5.8, soit iii) 6 threads, pour soutenir ce débit maximal. Ceci est cohérent avec la règle conservatrice qui suggère 1 thread par ressource (coeur ou disque) et assure qu'avec 8 + 4 + 4 = 16 nous aurions assez de threads.

b) Un ordinateur doit traduire l'adresse symbolique www.polenord.com en adresse IP numérique. Il a accès à 2 serveurs DNS redondants dans son réseau local, et aux 13 serveurs DNS racine redondants. Il doit commencer par interroger les serveurs DNS du réseau local pour trouver l'adresse des serveurs de nom pour le domaine polenord.com. Ces 2 serveurs ont exactement le même contenu et ont 60% de chances de contenir l'adresse recherchée. Toutefois, ils sont peu fiables et chaque serveur a 45% de chances d'être fonctionnel. Si ceux-ci sont en panne, ou n'ont pas ces adresses dans leur cache, l'ordinateur doit interroger les 13 serveurs racine. Ces 13 serveurs ont chacun l'information recherchée. Toutefois, ils sont eux aussi très peu fiables en ce moment en raison d'une attaque en déni de service, et chaque serveur a une probabilité de 10% d'être fonctionnel. Ensuite, ayant l'adresse des serveurs de nom pour le domaine polenord.com, l'ordinateur pourra obtenir l'adresse du site www.polenord.com. i) Quelle est la probabilité que l'ordinateur puisse obtenir l'adresse des serveurs de noms de polenord.com auprès des 2 serveurs DNS de son réseau local? ii) Quelle est la probabilité que les 13 serveurs DNS racine ne soient pas disponibles? iii) Quelle est la probabilité que l'ordinateur ne puisse obtenir l'information à propos des serveurs DNS pour le domaine polenord.com? **(2 points)**

Les 2 serveurs DNS locaux seront non disponibles avec une probabilité de $(1-.45)^2 = 0.3025$. *S'ils sont disponibles,* $1-0.3025 = 0.6975$, *ils auront la réponse dans 60% des cas, soit i)* $0.6975 \times 0.6 = 0.4185$. *Les 13 serveurs racines seront indisponibles si les 13 sont simultanément non fonctionnels ii)* $(1-0.1)^{13} = 0.254186583$. *L'ordinateur n'obtiendra pas sa réponse si elle*

n'est pas disponible sur le réseau local $1 - 0.4185 = 0.5815$ *et les serveurs racine ne sont pas disponibles* 0.254186583, *ce qui donne iii)* $0.5815 \times 0.254186583 = 0.147809498$.

Question 2 (5 points)

a) Un ordinateur A envoie un message à B à 13h00m10.200s pour obtenir le temps et reçoit une réponse à 13h00m10.600s, ces deux temps étant mesurés avec l'horloge de A. L'ordinateur B reçoit la requête de A à 13h00m05.300s et retourne sa réponse à A à 13h00m05.550s, ces deux temps étant mesurés avec l'horloge de B. Quel est le décalage à appliquer sur A? Quel est l'intervalle d'incertitude associé? **(2 points)**

a = 13h00m05.300s - 13h00m10.200s = -4.900s
b = 13h00m05.550s - 13h00m10.600s = -5.050s
Ajustement = (a+b)/2 = (-4.900s - 5.050s)/2 = -4.975s
Précision = (a-b)/2 = (-4.900s + 5.050s)/2 = 0.075s

Le décalage à appliquer à A est de -4.975s et l'incertitude est de +/- 0.075s.

b) Dans le nord du Canada, un serveur central d'exclusion mutuelle maintient les verrous qui servent à protéger les K fichiers qui représentent la liste de cadeaux de chaque personne. Ce serveur de verrous doit servir jusqu'à N threads répartis dans différents noeuds qui doivent accéder à ces fichiers. Chaque thread acquiert un après l'autre un certain nombre de verrous, suivant une discipline de verrouillage strict à deux phases, puis les relâche tous après la fin de la transaction. Lorsqu'un thread demande un verrou, cette opération est bloquante, le thread ne continue qu'après avoir obtenu le verrou. On suppose que les demandes de verrou sont toujours ordonnées correctement et qu'il n'y a pas d'interblocage. i) Combien de messages sont requis pour l'acquisition d'un verrou par un thread auprès du serveur central d'exclusion? ii) Si une case mémoire est requise dans le serveur pour chaque verrou en utilisation (numéro du thread qui détient le verrou) et pour chaque demande de verrou en attente (numéro du thread bloqué en attente pour ce verrou dans une liste), quel est le nombre maximal de cases mémoire qui peut être requis dans le serveur dans le pire cas? **(2 points)**

i) L'acquisition d'un verrou demande un message au serveur central par le thread et un message de réponse du serveur central vers le thread. Des accusés de réception pourraient être utilisés afin de ne pas bloquer indéfiniment en cas de perte d'un des deux messages sur le réseau. Il serait aussi possible de ne pas utiliser systématiquement d'accusé de réception mais plutôt que le thread envoie un message de vérification que sa demande a bien été reçue et est en attente si le délai est trop long. ii) Chaque verrou ne peut être détenu que par un seul thread. Chaque thread ne peut être en attente que d'un seul verrou. Ainsi, le nombre de cases mémoire pour les verrous en utilisation est au maximum de K et le nombre de cases pour les verrous en attente est au maximum de N (en fait N-1 puisqu'autrement on est en interblocage si tous les threads sont bloqués). Le nombre maximal de cases mémoire occupées pour cela est donc au total de K+N.

c) Un groupe d'ordinateurs utilise l'élection hiérarchique. Lorsqu'un des ordinateurs du groupe ne réussit pas à contacter le serveur, il déclenche le processus d'élection. Quelles sont les étapes qu'il va effectuer et quels messages va-t-il envoyer à cet effet? **(1 point)**

L'ordinateur qui ne réussit pas à contacter le serveur courant déclenche une élection. Il commence par envoyer un message d'élection au serveur le plus prioritaire. S'il ne reçoit pas de réponse après un certain temps (et possiblement plus d'un essai), il contacte le second serveur le plus prioritaire, et continue ainsi jusqu'à ce qu'il reçoive une réponse ou qu'il ne reste plus de serveur plus prioritaire que lui. Si aucun serveur plus prioritaire ne lui a répondu, il s'autoproclame le serveur et envoie un message à cet effet à tous.

Question 3 (4 points)

a) Lesquelles des transactions T, U, V et W pourraient être validées si une validation en reculant était utilisée pour vérifier la cohérence des transactions? Une validation en avançant? **(2 points)**

```
T: Début
U: Début
T: Read(a)
T: Read(b)
T: Read(c)
U: Read(c)
U: Read(d)
V: Début
V: Read(e)
T: Write(b)
T: Write(f)
W: Début
W: Read(e)
T: Compléter
W: Read(f)
U: Write(c)
U: Write(d)
U: Compléter
V: Read(d)
W: Write(a)
W: Write(b)
W: Compléter
V: Read(g)
V: Write(e)
V: Compléter
```

En reculant, au moment de compléter T, il n'y a pas de problème. Pour compléter U, il faut vérifier ce qu'il a lu (c, d) versus ce qui a été écrit par les transactions précédentes concurrentes, T (b, f). Il n'y a pas d'intersection et U peut compléter. Pour W, ses lectures (e, f) sont vérifiées versus les écritures de T (b, f) et U (c, d). Il y a intersection (f) et la transaction W doit être abandonnée. Pour V, les lectures (d, e, g) doivent être vérifiées versus les écritures de T (b, f) et U (c, d), mais pas de W qui a été abandonné. Il y a intersection (d) et la transaction V doit être

abandonnée. En fait, T était complété avant que W ne lise f, et U était complété avant que V ne lise d, mais cette information plus détaillée n'est pas utilisée par la méthode de base en reculant. Elle serait utilisée par la méthode avec les estampilles de temps.

En avançant, il faut vérifier ce qui est écrit par T (b, f) versus ce que U (c, d), V (e) et W (e) ont lu à ce moment. Il n'y a pas d'intersection et T peut compléter. Au moment de compléter U, ses écritures (c, d) sont comparées aux lectures à ce moment de V (e) et W (e, f); les lectures à venir de V et W ne sont pas encore connues à ce moment et n'ont donc pas d'impact sur la cohérence. Il n'y a pas d'intersection et la transaction U est validée. Pour W, ses écritures (a, b) sont comparées aux lectures de V (d, e) à ce moment. Il n'y a pas d'intersection et W peut compléter. La dernière transaction, V, complète nécessairement et ainsi les quatre transactions sont validées.

b) Trois transactions concurrentes, T, U et V veulent effectuer les opérations suivantes. Les opérations pour chacune des transactions sont effectuées en séquence, dans l'ordre. Cependant, l'ordre entre les opérations des trois transactions peut varier, puisque chaque transaction est effectuée par un thread parallèle concurrent. La cohérence des opérations est assurée par des verrous ordinaires (i.e. un verrou est pris sur une variable par la transaction au premier accès à cette variable, indifféremment pour une lecture ou une écriture). Est-ce qu'un interblocage pourrait se produire? Si oui, donnez une séquence d'opérations qui mènerait à un interblocage. **(2 points)**

	Transaction T	Transaction U	Transaction V
0	Début	Début	Début
1	Read(a)	Read(c)	Read(d)
2	Read(b)	Read(d)	Read(e)
3	Read(f)	Write(c)	Read(f)
4	Write(c)	Write(c)	Write(e)
5	Write(f)	Write(d)	Compléter
6	Compléter	Compléter	

Oui, un interblocage est possible. Si T effectue ses opérations 0 à 3, il acquiert les verrous pour a, b et f. Ensuite, U avec ses opérations 0 à 1 pourrait acquérir c. Finalement, V pourrait effectuer ses opérations 0 à 2 et acquérir d et e. A ce moment, T voudra acquérir c (détenu par U), U voudra acquérir d (détenu par V) et V voudra acquérir f, détenu par T et nous avons un interblocage.

Question 4 (4 points)

a) Deux scénarios sont comparés pour offrir un service de base de données. Dans le premier scénario a), 2 serveurs redondants viennent chacun avec une unité de disque RAID (4 disques dont au moins 3 sur 4 doivent être fonctionnels). Dans le second scénario b), 3 serveurs redondants viennent chacun avec 3 disques (qui doivent les 3 être fonctionnels). La probabilité d'être fonctionnel est 0.85 pour un serveur (hormis les disques) et de 0.9 pour un disque. Quel scénario sera le plus fiable? **(2 points)**

L'unité RAID a une probabilité de fonctionner de $0.9^4 = 0.6561$ *(4 disques fonctionnels) et* $4!/((4-3)!3!) \times 0.9^3 \times (1-0.9)^{4-3} = 0.2916$ *(exactement 3 disques fonctionnels) pour un total de* $0.2916 + 0.6561 = 0.9477$. *Pour le scénario a) un serveur est opérationnel si le serveur est fonctionnel de même que son unité RAID, ce qui donne* $0.9477 \times 0.85 = 0.805545$. *Le service sera disponible sauf si les 2 serveurs sont en panne, ce qui donne* $1-(1-0.805545)^2 = 0.962187253$. *Pour le scénario b), le serveur est fonctionnel si tout est fonctionnel (3 disques et le serveur)* $0.9^3 \times 0.85 = 0.61965$. *Le service sera foncitonnel sauf si les 3 serveurs sont en panne, ce qui donne* $1-(1-0.61965)^3 = 0.94497624$. *Ainsi, le scénario a) est un peu plus fiable.*

b) Dans le travail pratique 3, un gabarit `Heat`, `autoscaling.yaml`, vous était fourni en exemple. Quel est le principe d'un service qui se met à l'échelle automatiquement sur `OpenStack` avec `Heat`? La section suivante est extraite de ce fichier. Expliquez à quoi sert cette section. **(2 points)**

```
cpu_alarm_high:
  type: OS::Ceilometer::Alarm
  properties:
    description:
    meter_name: cpu_util
    statistic: avg
    period: 60
    evaluation_periods: 1
    threshold: 50
    alarm_actions:
      - {get_attr: [web_server_scaleup_policy, alarm_url]}
    matching_metadata: {'metadata.user_metadata.stack':
                          {get_param: "OS::stack_id"}}
    comparison_operator: gt
```

La mise à l'échelle se base sur un répartiteur de requêtes, et un bassin de machines virtuelles identiques pouvant servir les requêtes. Lorsque la charge est trop élevée, de nouvelles instances de machines virtuelles sont ajoutées au bassin. Lorsque la charge est trop faible, des machines virtuelles sont retirées. Ainsi, le nombre de machines virtuelles disponibles s'ajuste en fonction de la charge. Le gabarit `Heat` *permet de spéficier les différentes ressources requises pour implémenter un tel service, ainsi que leurs paramètres. La section de gararit* `Heat` *fournie sert à instancier une ressource de type* `OS::Ceilometer::Alarm`. *Cette ressource monitore la métrique* `cpu_util` *qui est le taux d'utilisation du CPU. Si le taux d'utilisation du CPU monte au-dessus de 50% pendant 60 secondes, l'alarme exécute une action qui active la ressource* `web_server_scaleup_policy` *pour ajouter des machines virtuelles.*

Question 5 (3 points)

Vous devez planifier un nouveau centre de données et comparer différents scénarios. Vous avez déjà prévu examiner la performance des systèmes (et les revenus qu'on peut en tirer), le coût des ordina-

teurs, le coût du bâtiment ainsi que les coûts d'opération et de renouvellement des équipements, afin de déterminer le projet le plus rentable sur la durée de vie anticipée. Devez-vous comme ingénieur aussi vérifier les aspects du développement durable de ce projet? i) Quelles sont les lois applicables? ii) Quelles sont les différentes phases du cycle de vie? iii) Quels sont les quatre différents types d'impact sur l'environnement? **(3 points)**

L'ingénieur doit tenir compte des conséquences de l'exécution de ses travaux sur l'environnement dans une perspective de développement durable. i) Ceci est prescrit par la loi canadienne de 2008 sur le développement durable, la loi québécoise de 2006 sur le développement durable, et le code de déontologie de l'Ordre des Ingénieurs du Québec. La procédure d'évaluation environnementale du BAPE ne s'applique normalement pas à un centre de données. ii) Les phases du cycle de vie sont la fabrication (construction du site et fabrication des équipements), le transport (des matériaux pour la construction et des équipements), l'opération du site (entretien, alimentation électrique...) et finalement la fin de vie (la démolition ou conversion du bâtiment, la restauration du site, et le recyclage ou la mise aux rebuts des matériaux et équipements). iii) Pour toutes les activités qui se retrouvent dans ces différentes phases, il faut examiner l'impact sur la santé humaine, sur l'écologie, sur les changements climatiques et sur l'appauvrissement des ressources. Pour un centre de données typique, une grande partie de l'impact est reliée à la consommation d'énergie pendant l'opération de l'équipement informatique et de la climatisation. Un autre impact non négligeable est la fabrication des équipement informatiques et électriques, en particulier le raffinage de l'or et du cuivre requis pour ces équipements. Etant donné l'importance de la consommation électrique, une source d'énergie propre, et un climat froid qui ne requiert pas de climatisation, peuvent diminuer considérablement l'impact négatif d'un centre de données.

Le professeur: Michel Dagenais