**ECOLE POLYTECHNIQUE DE MONTREAL**

**Département de génie informatique et génie logiciel**

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

---

# CORRIGÉ DU CONTRÔLE PÉRIODIQUE

**DATE: Vendredi le 23 février 2018**

**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 serveur reçoit des requêtes qui se font en 7 étapes: 1) ouverture d'une connexion TCP, réception d'une commande et envoi de la réponse en 10ms de CPU, 2) attente de 20ms, 3) réception d'une seconde commande qui prend aussi 10ms de CPU à traiter, 4) attente de 20ms, 5) réception d'une troisième commande qui prend aussi 10ms de CPU à traiter, 6) attente de 20ms, 7) fermeture de la connexion TCP, ce qui indique la fin de la requête. Si le serveur s'exécute avec un seul thread d'exécution qui traite séquentiellement les requêtes, combien de requêtes par seconde peut-il traiter? Si le serveur exécute de nombreux threads qui se partagent les requêtes, mais roulent sur un seul coeur de CPU, combien de requêtes par seconde peut-il traiter? **(2 points)**

*Chaque requête prend un total de 10+20+10+20+10+20=90ms, un seul thread peut donc en traiter 1000ms/90ms = 11.11 par seconde. Toutefois, puisque chaque requête ne prend que 30ms de CPU, avec plusieurs threads on peut en traiter 1000ms/30ms = 33.33 par seconde.*

b) Un serveur doit transmettre son contenu pour le répliquer sur 2 autres serveurs, soit 1000000 paquets de 4096 octets. La transmission d'un petit paquet comme un accusé de réception occupe le réseau pendant 0.1ms alors qu'un paquet de 4096 octets l'occupe pendant 0.5ms. On suppose que le nombre de paquets perdus sur ce réseau est négligeable. i) Quel est le temps requis pour un envoi séparé vers chacun des 2 autres serveurs avec un accusé de réception pour chaque paquet envoyé? Quel est le temps requis pour un envoi simultané en multi-diffusion aux 2 serveurs avec un accusé de réception négatif à la toute fin (sur réception de la fin de fichier, le récepteur envoie la liste des paquets manquants)? **(2 points)**

*Dans le premier cas, il faut l'envoi de 1000000 paquets pour le contenu, à 0.5ms chacun, et de 1000000 paquets d'accusé de réception, à 0.1ms chacun, vers chacun des deux serveurs. Le total est donc de* $2 \times 1000000 \times (0.5ms + 0.1ms) = 1200s$. *Dans le second cas, on envoie 1000000 paquets à 0.5ms chacun en multi-diffusion aux deux serveurs et il n'y a pas d'accusés de réception négatif si aucun paquet n'est perdu. Le temps est donc de* $1000000 \times 0.5ms = 500s$.

c) Lors d'un envoi de message de groupe atomique, un noeud a reçu le message et est en attente de la confirmation que tous l'ont reçu avant de livrer le message à l'application. Que pourra faire ce noeud après un certain délai i) si le message de confirmation du noeud d'envoi a été perdu? ii) Si le noeud d'envoi a flanché dans le milieu de l'envoi de ses messages de confirmation? **(1 point)**

*Au bout d'un certain temps, le noeud va interroger le noeud d'envoi pour savoir ce qui en est. Si le message de confirmation s'était perdu, la réponse qui ne lui était pas parvenue sera retransmise et à sa réception il pourra livrer le message de groupe atomique à l'application. Si le noeud d'envoi a flanché, il ne répondra pas à cette interrogation. Le noeud en réception ne peut donc savoir s'il faut livrer le message atomique à l'application ou au contraire le détruire. Il faut que le système ait une manière de récupérer de cette erreur. Par exemple, le noeud d'envoi pourrait avoir écrit dans un journal le fait qu'il commençait à envoyer des confirmations. Après avoir flanché puis redémarré, il pourra relire ce journal et savoir quoi répondre au client récepteur qui l'interroge. Le noeud en réception peut aussi interroger les autres membres du groupe. Si un*

*des autres membres a reçu une confirmation, le noeud en réception peut se baser là-dessus pour livrer le message à l'application.*

# Question 2 (5 points)

a) Un client accède un service de fichiers directement par le biais d'une librairie générée par gRPC. La fonction RPC pour lire le contenu d'un fichier est `FileRead` tel que montré ci-après. i) Si on effectue un appel avec comme arguments 1000 pour le `fileId` et la valeur par défaut pour le `fileOffset`, et que le contenu retourné est le contenu complet du fichier qui contient 300 octets, expliquez combien d'octets sont requis pour encoder ces 2 messages (FileSegment et FileContent). ii) On ajoute un argument optionnel, `optional uint64 length = 3 [default = 0];`, au message FileSegment et le serveur est mis à jour en conséquence pour lire tout le fichier si la longueur est 0, et se limiter à la longueur spécifiée autrement. Est-ce que cela va continuer à fonctionner de manière transparente pour un ancien client qui ne spécifie pas de longueur? Expliquez. **(2 points)**

```
message FileSegment {
  required uint64 fileId = 1;
  optional uint64 fileOffset = 2 [default = 0];
}
message FileContent {
  required bytes content = 1;
}
service FileService {
rpc FileRead (FileSegment) returns (FileContent) {}
...
}
```

*Pour fileId, le numéro de champ et le type peuvent entrer dans un seul octet. La valeur 1000 requiert 2 octets avec les varint utilisés. Pour fileOffset, aucune valeur n'est spécifiée et le champ peut être omis. Un total de 3 octets est donc requis. Pour le contenu retourné, nous avons content avec 1 octet pour le champ/type, 2 octets pour la longueur (300) en varint, et 300 octets pour le contenu lui-même. La longueur totale pour FileContent est donc de 303. Le grand total est de 306 octets. Si un argument optionnel est ajouté à la fonction, cela fonctionnera correctement si la valeur par défaut correspond au comportement de la version précédente de la fonction. C'est le cas ici puisque la valeur par défaut est de 0 (lire tout le fichier comme avant cette option). Ainsi, un ancien client qui ne spécifie pas le nouveau champ verra exactement le même résultat qu'avant le changement sur le serveur.*

b) Une application de dessin répartie est programmée avec Java RMI. Deux organisations possibles sont évaluées. Dans l'organisation A, l'objet DessinA est un objet réseau qui contient un vecteur d'objets réguliers FormeA qui sont des formes géométriques. Dans la seconde organisation, B, l'objet DessinB est un objet réseau qui contient un vecteur d'objets réseau FormeB qui sont des

formes géométriques. Un client doit lire le contenu d'un dessin afin de l'afficher (appel pour obtenir le vecteur de formes du dessin, puis appel pour chaque forme pour lire son contenu afin de l'afficher). Si le dessin D1 contient 10 formes géométriques, F1 à F10, répondez à i) et ii) pour chacune des deux organisations, A et B. i) Combien d'appels réseau seront requis pour ainsi afficher le dessin? ii) Quel sera le contenu de la table des objets exportés et de la table des objets importés dans le client et dans le serveur? **(2 points)**

*Avec l'organisation A, un appel réseau est requis afin d'obtenir le vecteur de formes. On suppose que le proxy vers D1 est déjà dans le client, puisque la manière de l'obtenir n'est pas spécifiée. Le vecteur et les formes qu'il contient sont des objets locaux et leur accès ne demande pas d'appel réseau. La table des objets exportés du serveur contient D1 et la table des objets importés du client contient le proxy de D1. Avec l'organisation B, il faut 1 appel pour obtenir le vecteur de formes et un appel réseau pour obtenir le contenu de chaque forme, soit un total de 11. La table des objets exportés du serveur contient D1 et F1 à F10 alors que la table des objets importés du client contient les proxy de D1 et F1 à F10.*

c) Dans votre travail pratique, expliquez ce que fait la fonction `lock(nom, clientid, checksum)` selon les différents cas possibles. **(1 point)**

*Cette fonction verrouille le fichier* nom *sur le serveur pour le client* clientid, *s'il existe et s'il n'est pas déjà verrouillé par un autre client. De plus, si la version la plus récente n'est pas présente sur le client (la somme de contrôle ne correspond pas à celle envoyée,* checksum*), le fichier est copié dans le répertoire local sur le client.*

# Question 3 (5 points)

a) Sur OpenStack, le service Nova s'occupe de faire exécuter les machines virtuelles sur les noeuds disponibles. Lorsqu'un client demande de démarrer une nouvelle machine virtuelle, comment Nova décide sur quel noeud démarrer la nouvelle machine virtuelle? **(2 points)**

*Nova regarde la liste des noeuds présents et fonctionnels. Parmi ceux-ci, il détermine le sous-ensemble éligible des noeuds qui satisfont aux contraintes de la machine virtuelle à créer (répertoire d'instructions, mémoire, capacité...). Parmi les noeuds éligibles, il choisit le noeud de plus forte priorité selon différents critères qui peuvent être configurés (charge, nombre de VM déjà présentes, puissance du noeud...).*

b) Une machine virtuelle contient une image de 2000000 pages de 4096 octets. On veut migrer cette machine virtuelle d'un noeud à un autre. L'envoi d'une page de 4096 octets sur le réseau prend 0.1ms. Quel sera le temps requis pour la migration si i) la machine virtuelle est peu occupée et modifie 10 pages par seconde? ii) Si la machine virtuelle est très occupée et modifie 10000 pages par seconde? **(2 points)**

*La première ronde de copie prend* $2000000 \times 0.1ms = 200s$. *Si la machine virtuelle modifie 10 pages par secondes, 2000 pages seront modifiées pendant ce temps, ce qui prendra 0.2s à transmettre. Pendant cette seconde copie, 2 pages seront modifiées, ce qui prend .2ms à transmettre et il n'y aura vraisemblablement plus de pages modifiées rendu là. De toutes manières, lorsque le nombre de pages restant est très faible, on arrête la machine virtuelle pendant la dernière ronde*

*de copie. La migration prendra donc environ 200s+.2s+.2ms=200.2002s et l'arrêt de la machine virtuelle sera très bref. Si la machine virtuelle modifie 10000 pages par seconde, pendant la première copie de 200s, il y aura* $200s \times 10000p/s = 2000000$ *pages modifiées. On constate donc qu'il n'y a aucun progrès puisque les pages sont modifiées aussi vite qu'elles sont copiées. La migration ne terminera jamais dans ces conditions. Elle ne pourra pas se faire sans arrêter la machine virtuelle durant toute la copie.*

c) Kubernetes et Docker sont deux systèmes pour exécuter des conteneurs. Expliquez le rôle de chacun dans une solution infonuagique basée sur les conteneurs. **(1 point)**

*Docker est un format d'image et un environnement pour exécuter un conteneur. Sa force est de simplifier la construction des images et de faciliter l'interface pour démarrer, exécuter et arrêter une telle image sur différents systèmes (incluant Linux et Windows). Kubernetes est un système d'orchestration des conteneurs. On peut spéficier comment déployer les conteneurs, incluant la réplication et la répartition de requêtes. Les conteneurs déployés seront souvent des images Docker avec l'environnement d'exécution associé.*

## Question 4 (5 points)

a) Un serveur NFS dessert de nombreux clients. Les clients effectuent en moyenne 1 écriture et 10 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 90% des cas. Parmi les blocs en cache, 40% 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, 30% ont été modifiés et nécessitent une lecture en plus, alors que 70% sont valides. L'écriture d'un bloc sur le serveur prend 15ms de disque. La lecture d'un bloc du serveur prend 10ms de disque dans 20% des cas, et est servie à partir de la cache d'entrée-sortie en temps négligeable dans 80% des cas. Une validation d'un bloc du serveur prend 10ms de disque dans 10% des cas, et est servie à partir de la 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 8 disques et que les requêtes sont réparties uniformément entre les disques? **(2 points)**

*Pour chaque lecture d'un client, on a .1 lecture sur le serveur (non présent en cache),* $.9 \times .4 = .36$ *rien (bloc en cache et récent),* $.9 \times .6 \times .7 = .378$ *revalidation sur le serveur (copie en cache validée),* $.9 \times .6 \times .3 = .162$ *revalidation sur le serveur et en plus lecture sur le serveur (copie en cache non validée). Pour 10 lectures sur le client, on a donc* $10 \times (.1 + .162) = 2.62$ *lectures et* $10 \times (.378 + .162) = 5.4$ *revalidations sur le serveur à chaque seconde, en plus de 1 écriture par seconde.*

*Le disque sur le serveur subit donc pour un client la charge suivante par seconde* $1\ ecriture \times 15ms + 2.62\ lectures \times .2 \times 10ms + 5.4\ revalidations \times 0.1 \times 10ms = 25.64ms$. *Ainsi, sur 8 disques on peut soutenir* $8 \times 1000ms/25.64ms = 312.01$ *clients.*

b) Un service de fichiers CODA est répliqué sur 4 serveurs (i.e. chaque fichier se retrouve en 4 copies). Chaque serveur possède 6 disques. Chaque disque peut effectuer 100 accès (lecture ou écriture) par seconde. Les clients, lors des ouvertures ou fermetures de fichiers, font des 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)? **(2 points)**

*En lecture, les serveurs opèrent en parallèle. Avec 4 serveurs de 6 disques à 100 accès/s, on a une capacité maximale de* $4 \times 6 \times 100 = 2400$ *lectures par seconde. En écriture, chaque écriture doit être faite sur les 4 serveurs. On peut donc soutenir 4 fois moins d'écritures que de lectures soit 600 écritures pas seconde.*

c) Quel est le principe de fonctionnement de BitTorrent? En quoi est-ce optimisé pour les gros fichiers? **(1 point)**

*Avec BitTorrent, les fichiers sont offerts en morceaux. De plus, les morceaux les plus rares sont offerts en priorité. Ceci permet de rapidement mettre en circulation tous les morceaux d'un fichier et de paralléliser le plus rapidement possible les transferts, avec de nombreux clients qui participent au transfert. Ceci est particulièrement intéressant pour les gros fichiers. En effet, pour un gros fichier, s'il faut attendre une copie complète avant qu'un client puisse participer à la retransmission, cela prend beaucoup de temps.*

Le professeur: Michel Dagenais