

H24 :**Question 1 :**

En ignorant les problèmes gênants tels que les pares-feux pouvant bloquer certains protocoles, indiquez pour les applications suivantes s'il serait généralement préférable d'utiliser UDP ou TCP.

- Diffusion d'une vidéo en direct sur Internet : **UDP**
- Messagerie instantanée / courriel : **TCP**
- Connexion au site Web de votre banque : **TCP**
- Voix sur IP : **UDP**
- Transferts de fichiers volumineux : **TCP**
- Recherche d'une valeur (de taille très faible) à partir d'un service d'annuaire : **UDP**

Question 2 :

Dans un conteneur docker, le noyau du système d'exploitation est: Veuillez choisir une réponse.

- Celui qui est installé dans la première couche (first layer) du conteneur
- Celui du système qui exécute la commande "docker run"
- Celui qui est installé dans le conteneur
- Celui du système hôte ✓**

Question 3 :

RPC tente de faire en sorte que les appels de procédures distantes aient le même aspect que les appels de procédures locales. Mais l'illusion n'est pas parfaite. Sélectionnez toutes les options décrivant correctement les différences entre une fonction locale et un RPC.

- Veuillez choisir au moins une réponse.
- a. Les appels RPC nécessitent un paramètre supplémentaire pour identifier le serveur ✓
 - b. Les appels RPC ont des erreurs différentes (plus nombreuses) ✓
 - c. Les appels RPC sont limités à l'appel par valeur
 - d. Les appels RPC sont facturés comme des appels interurbains
 - e. Les appels RPC peuvent avoir une latence plus élevée ou variable ✓

f. Les appels de fonctions locales nécessitent un malloc d'un objet dans le tas pour représenter le pointeur de retour

Question 4 :

Parmi les éléments suivants concernant les interfaces SOAP et REST, sélectionnez ceux qui sont vrai. Veuillez choisir au moins une réponse.

- a. SOAP doit utiliser XML pour coder ses données. ✓
- b. REST est indépendant de l'architecture, ce qui n'est pas le cas de SOAP.
- c. REST doit utiliser JSON pour coder ses données.
- d. REST envoie et reçoit du contenu pendant que SOAP utilise des appels de procédure à distance.
- e. Les opérations à effectuer sont codées dans le document avec SOAP mais dans l'URL avec REST. ✓

Question 5 :

Kubernetes est un système d'orchestration de containers d'applications sur des grappes de serveurs composées: Veuillez choisir au moins une réponse.

- a. de <>pods<> qui sont des groupes d'un ou plusieurs containers d'applications et qui partagent des volumes partagés et une même adresse IP ✓
- b. de <>pods<> qui sont des machines qui coordonnent la planification et la gestion des conteneurs d'applications sur le cluster
- c. de <>nodes<>, groupe de pods travaillant ensemble
- d. de <>nodes<> où sont déployés les applications ✓
- e. de <>nodes<> qui sont des groupes d'un ou plusieurs containers d'applications et qui partagent des volumes partagés et une même adresse IP

Question 6 :

Soit un système de fichiers NFS monté sur un système Linux. Lorsqu'un processus P essaie d'accéder à un fichier situé dans ce système de fichiers, un appel RPC est émis. Si cet appel expire (timeout), quelle est l'action effectuée par le module client NFS.

- a. NFS ne retourne pas le contrôle au processus P, mais reste bloqué.
- b. NFS retourne le contrôle au processus P.

c. NFS va refaire l'appel RPC. ✓

d. Le processus P continu à s'exécuter sans attendre la réponse de NFS.

Question 7 :

Un message (le contenu d'une structure de données) doit être envoyé comme argument pour un appel de procédure à distance. Ce message contient trois champs qui sont des chaînes de caractères, dont les nombres de caractères pour cette instance spécifique du message sont respectivement de 18, 463 et 160, et trois champs qui sont des entiers, dont les valeurs pour cette instance spécifique du message sont respectivement de 32, 134 et 23376.

Combien d'octets seront requis pour encoder ces 6 champs avec CORBA CDR sur un ordinateur 32 bits?

Réponse : Calcul pour encodage CORBA CDR sur 32 bits:

Chaînes de caractères avec padding (ça doit être des multiples de 4):

1er champ : 4 octets + 18 octets = 22; 22 // 4 = 2; 22 + 2 = 24

2e champ : 4 octets + 463 octets = 467; 467 // 4 = 1; 467 + 1 = 468

3e champ : 4 octets + 160 octets = 164; 164 // 4 = 0; 164 + 0 = 164

Chaque entier = 4 octets, donc 3 x 4 = 12

Total: 24 + 468 + 164 + 12 = 668 octets

Question 7 (Alternate) :

Un message (le contenu d'une structure de données) doit être envoyé comme argument pour un appel de procédure à distance. Ce message contient trois champs qui sont des chaînes de caractères, dont les nombres de caractères pour cette instance spécifique du message sont respectivement de 18, 310 et 118, et trois champs qui sont des entiers, dont les valeurs pour cette instance spécifique du message sont respectivement de 80, 188 et 21539. Combien d'octets seront requis pour encoder ces 6 champs avec CORBA CDR sur un ordinateur 32 bits?

Réponse : Calcul pour encodage CORBA CDR sur 32 bits:

1er champ : 4 octets + 18 octets = 22; 22 // 4 = 2; 22 + 2 = 24

2e champ : 4 octets + 310 octets = 314; 314 // 4 = 2; 314 + 2 = 316

3e champ : 4 octets + 118 octets = 122; 122 // 4 = 2; 122 + 2 = 124

Chaque entier = 4 octets, donc 3 x 4 = 12

Total: 24 + 316 + 124 + 12 = 476 octets

Question 8 :

Pour les différents services infonuagiques comme Amazon EC2, on parle de stockage d'instance, de stockage de bloc (EBS), et de stockage d'objets (S3). Pour chacune des définitions suivantes, sélectionnez le type de stockage qui correspond.

Stockage d'instance : Son contenu est perdu lorsque l'instance est arrêtée.

Stockage de bloc : Son contenu persiste après l'arrêt de l'instance et il peut être accédé à nouveau par une nouvelle instance.

Stockage d'objets : Le stockage est comme une page Web qui supporte GET et PUT

Question 9 :

Pour chacune des définitions suivantes, sélectionnez le type de transparence qui correspond.

Permet aux utilisateurs et aux programmes d'effectuer leurs tâches même en cas de problèmes matériels ou logiciels : Transparence de défectuosité

Permet d'accéder aux ressources en ne connaissant que leur nom :

Transparence de localisation

Permet à plusieurs processus de fonctionner simultanément, en utilisant des ressources partagées, sans interférence entre eux :

Transparence de concurrence

Permet d'utiliser les ressources locales et distantes à l'aide d'opérations identiques : Transparence d'accès

Question 10 :

Un serveur de disques composé de disques SSD traite les requêtes de clients. 20 Mbps (mégabits/seconde) en moyenne sont requis par chaque client dans un réseau communiqué. Le serveur est connecté au réseau par une prise fournissant 1 Gbps (gigabit par seconde). Son bus a une capacité de 8 gigaoctets/s avec 6 disques connectés. Ces derniers fournissent 50 mégaoctets/s. Les chiffres fournis sont des puissances de 10 (e.g. mega 106 et giga = 109). Combien de clients peut-il supporter?

Réponse : Réseau :

Capacité : $1 \text{ Gbps} = 1 \times 10^9 \text{ bits/s} = 125 \times 10^6 \text{ octets/s}$

Par client : $20 \text{ Mbps} = 20 \times 10^6 \text{ bits/s} = 2.5 \times 10^6 \text{ octets/s}$

Nombre maximum de clients = $125 \times 10^6 / 2.5 \times 10^6 = 50$

Bus et disques :

Bus : $8 \text{ GB/s} = 8 \times 10^9 \text{ octets/s}$

Disques : $6 \times 50 \text{ MB/s} = 300 \times 10^6 \text{ octets/s}$

→ Ces capacités dépassent le débit du réseau

→ La limite est donc le réseau, soit 50 clients.

Question 11 :

Un processus serveur reçoit des requêtes de clients par le biais d'appels de méthode à distance. Le serveur reçoit 30 requêtes par seconde et chaque requête crée un nouvel objet réseau de type session qui sera utilisé pendant 180 secondes. On envisage une stratégie pour déterminer quand les objets réseau peuvent être libérés. Pour cette stratégie, une notification est envoyée par le client lorsque l'objet n'est plus utilisé. Cependant, on estime que pour 2% des requêtes, le message de notification ne parviendra pas au serveur et ainsi l'objet ne sera pas libéré et restera en mémoire dans le serveur. Pour cette raison, le serveur est redémarré au milieu de chaque nuit afin de repartir à 0 et que les objets ne s'accumulent pas d'un jour à l'autre. Quel est le nombre d'objets qui n'ont pas été libérés à la fin de la journée ?

Réponse : Requêtes par jour: $30 \text{ req/s} \times 60 \text{ s} \times 24 \text{ h} = 2592000 \text{ requêtes/jour}$

Requêtes avec notification perdue (2%): $2592000 \times 0.02 = 51840 \text{ objets non libérés}$

Question 12 :

Quel est le nombre de requêtes qui ne seront pas finalisées par le serveur à cause du redémarrage ?

Réponse : Chaque requête crée un objet qui reste actif pendant 180s. Requêtes actives à tout moment: $30 \text{ req/s} \times 180s = 5400 \text{ requêtes}$

Lors du redémarrage, ces 5400 requêtes en cours ne seront pas finalisées.

Question 13 :

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 80% des cas. Parmi les blocs en cache, 70% 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 25ms 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 d'écritures sur des blocs que chaque client demande au serveur par seconde ?

Réponse : Les écritures sont toujours envoyées au serveur, donc 2 écritures par seconde.

Question 14 :

Quelle est la probabilité qu'un bloc se trouve en cache et a été validé depuis moins de 3s ? Donnez la réponse avec deux décimales.

Réponse : Probabilité qu'un bloc se trouve en cache: 0.8

Probabilité que parmi ces blocs,

Il ait été validé depuis moins de 3s: 0.7

Probabilité totale: $0.8 \times 0.7 = 0.56$

Question 15 :

Quelle est la probabilité qu'un bloc se trouve en cache mais on doit demander une validation auprès du serveur et la validation soit positive?

Réponse: Probabilité qu'un bloc se trouve en cache: 0.8

Probabilité que parmi ces blocs, il n'ait PAS été validé depuis moins de 3s: 1 - 0.7 = 0.3

Probabilité que parmi ces blocs non validés, la validation soit positive: 0.6

Probabilité totale: $0.8 \times 0.3 \times 0.6 = 0.144$

Question 16 :

Quel est le temps de disque en moyenne pour la lecture d'un bloc sur le serveur ? Donnez la réponse en secondes.

Réponse: Temps moyen = $15 \text{ ms} \times 0.30 = 4.5 \text{ ms}$

En secondes : $4.5 \text{ ms} = 0.0045 \text{ s}$

Question 17 :

Quel est le temps de disque en moyenne qui prend une validation? Donnez la réponse en secondes. Donnez la réponse en secondes (avec quatre décimales).

Réponse: Temps moyen = $15 \text{ ms} \times 0.10 = 1.5 \text{ ms}$

En secondes (avec 4 décimales) : $1.5 \text{ ms} = 0.0015 \text{ s}$

Question 18 :

Quel est le temps en moyen qui prendre un client? Donnez la réponse en secondes. Donnez la réponse en secondes (avec cinq décimales).

Réponse : – Chaque écriture coûte 25 ms

→ Total écritures = $2 \times 25 \text{ ms} = 50 \text{ ms}$

Lectures :

– 80 % des lectures (soit 6,4 lectures) proviennent du cache

• Parmi ces 6,4, 70 % (4,48 lectures) n'ont pas besoin de validation (coût négligeable)

• Les 30 % restants (1,92 lectures) nécessitent une validation

– Coût de validation : $15 \text{ ms} \times 10 \% = 1.5 \text{ ms}$

• Parmi ces 1,92 validations, 40 % (0,768) nécessitent en plus une lecture (coût moyen de lecture = $15 \text{ ms} \times 30 \% = 4.5 \text{ ms}$)

– Coût total pour validations = $1.92 \times 1.5 \text{ ms} + 0.768 \times 4.5 \text{ ms} = 2.88 \text{ ms} + 3.456 \text{ ms} = 6.336 \text{ ms}$

– Les 20 % restants (1,6 lectures) ne sont pas en cache et nécessitent directement une lecture

– Coût = $1.6 \times 4.5 \text{ ms} = 7.2 \text{ ms}$

→ Total lectures = $6.336 \text{ ms} + 7.2 \text{ ms} = 13.536 \text{ ms}$

Temps disque moyen par client :

Total = $50 \text{ ms} + 13.536 \text{ ms} = 63.536 \text{ ms}$

En secondes (avec cinq décimales) : $63.536 \text{ ms} = 0.06354 \text{ s}$

Question 19 :

Quel est le nombre de clients maximal qui 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?

Réponse : Nombre maximal de clients = $(\text{Temps disque total disponible}) / (\text{Temps disque consommé par client}) = 16 / 0.06353 \approx 251.61 = 251$

Question 20 :

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 100 / seconde et le serveur peut traiter chaque requête en 8ms. Calculez N, le nombre moyen de requêtes dans le système

Réponse : – Temps de traitement par requête = 8 ms → $\mu = 1 / 0.008 = 125 / \text{s}$

Calcul de N (nombre moyen de requêtes dans le système)

Utilisation $\rho = \lambda / \mu = 100 / 125 = 0.8$

Pour un M/M/1 : $N = \rho / (1 - \rho) = 0.8 / 0.2 = 4$

Question 21 :

Calculez W le temps de réponse moyen en secondes.

Donnez la réponse en secondes (avec deux décimales). Utilisez la virgule (,) pour séparer la partie entière de la partie décimale.

Réponse : – $W = 1 / (\mu - \lambda) = 1 / (125 - 100) = 1 / 25 = 0.04 \text{ s}$

Ou – $W = N / \lambda = 4 / 100 = 0.04 \text{ s}$

Question 22 :

On prévoit ouvrir quatre nouvelles succursales, avec un nouveau serveur pour chacune des succursales qui recevra le même nombre de requêtes et aura la même capacité de traitement.

Quel sera le temps d'attente moyen W, en secondes, si chaque serveur a sa propre queue d'attente? Donnez la réponse en secondes (avec deux décimales).

Réponse : Chaque serveur reste identique → temps de réponse moyen par serveur = 0.04 s

Question 23 :

On prévoit ouvrir quatre nouvelles succursales, avec un nouveau serveur pour chacune des succursales qui recevra le même nombre de requêtes et aura la même capacité de traitement.

Quel sera le temps d'attente moyen W, en secondes, si une queue unique alimente les cinq serveurs? Donnez la réponse en secondes (avec cinq décimales).

Réponse: Enoncé : Une file unique alimente 5 serveurs identiques.

Pour l'ensemble :

– Taux total d'arrivée $\lambda_{\text{total}} = 5 \times 100 = 500 / \text{s}$

– Chaque serveur a $\mu = 125 / \text{s}$, $m = 5$ serveurs

– Utilisation globale par serveur $\rho = \lambda_{\text{total}} / (m \times \mu) = 500 / (5 \times 125) = 500 / 625 = 0.8$

Pour un système M/M/m, le temps de réponse moyen $W = W_q + 1 / \mu$, où W_q (le temps d'attente dans la file) s'obtient via la formule d'Erlang C.

Calcul détaillé :

– Calcul de A = $m \times \mu = 5 \times 125 = 625$

– Calcul de la probabilité d'attente P(wait)

– Dénominateur pour P_0 :

$\sum_{k=0}^{m-1} (A^k / k!) = |k=0 : 40\%| = 1 |k=1 : 4/1! = 4 |k=2 : 4^2/2! = 16 |k=3 : 4^3/3! = 64/6 = 10,6 |k=4 : 4^4/4! = 256/24 = 10,67$

→ Somme = $1 + 4 + 8 + 16,7 + 32,33 = 64,6$

Terme supplémentaire : $(A^m / m!) \times (1/(1-\rho)) = (4^5/5!) * (1/0,2)$

= $(1024/120) \times 5 \approx 8,5333 \times 5 = 42,67$

– Dénominateur total = $34,33 + 42,67 = 77$

– $P_0 = 1 / 77 \approx 0,01299$

– $P(\text{wait}) = (A^m / m!) \times (1/(1-\rho)) \times P_0 \approx 42,67 \times 0,01299 \approx 0,554$

Calcul de W_q (temps moyen d'attente dans la file)

Formule simplifiée : $W_q = P(\text{wait}) \times (1/\mu) / (m \times (1-\rho))$

Avec $1/\mu = 1/125 = 0,008 \text{ s}$, $m = 5$, et $(1-\rho) = 0,2$

$W_q = (0,554 \times 0,008) / (5 \times 0,2) = 0,004432 / 1 = 0,004432 \text{ s}$

Temps de réponse total $W = W_q + (1/\mu) = 0,004432 + 0,008 = 0,012432 \text{ s}$

Arrondi à cinq décimales : 0,01243 s

A24 :**Question 1 :**

Un message (le contenu d'une structure de données) doit être envoyé comme argument pour un appel de procédure à distance. Ce message contient trois champs qui sont des chaînes de caractères, dont les nombres de caractères pour cette instance spécifique du message sont respectivement de 18, 310 et 118, et trois champs qui sont des entiers, dont les valeurs pour cette instance spécifique du message sont respectivement de 80, 188 et 21539. Combien d'octets seront requis pour encoder ces 6 champs avec CORBA CDR sur un ordinateur 32 bits?

Réponse : Calcul pour encodage CORBA CDR sur 32 bits:

1er champ : 4 octets + 18 octets = 22; 22 // 4 = 2; 22 + 2 = 24

2e champ : 4

<p>ii) Ils peuvent supporter 1000 requêtes /s soit : $(1000r/s)/(10/client-s) = 100 clients$.</p> <p>iii) Si chaque requête reste dans le système 4ms CPU + 5ms disque = 9ms en moyenne, cela demande $(1000r/s) \times (1s/1000ms) \times (9ms-thread/r) = 9 threads$, soit <u>9 fils d'exécution</u>.</p> <p>b) Un service de fichiers prend 8ms 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 100 / seconde. i) Quel est le taux d'utilisation U du serveur. ii) Calculez N, le nombre moyen de requêtes dans le système. iii) Quel est W le temps de réponse moyen en secondes pour ce cas?</p> <p>Réponse : Le taux de service u est de $(1000ms/s)/(8ms/r) = 125r/s$ et le taux d'utilisation U = $l/u = (100r/s)/(125r/s) = 0.8$.</p> <p>i) En conséquence, $N = 0.8 / (1 - 0.8) = 4$</p> <p>ii) $W = N/l = (4)/(100r/s) = 0.04s$ ou 40ms</p> <p>c) Les téléphones intelligents peuvent exécuter de nombreuses applications. Il est important que chaque application ne puisse pas par défaut accéder à ses propres données. Quel mécanisme est utilisé à cette fin sur les systèmes Android? Sur les systèmes IOS?</p> <p>Réponse : Dans un cas comme dans l'autre, chaque application installée à son propre répertoire de données. Dans le cas de Android, chaque application se voit attribuer un identificateur d'usager différent. Le mécanisme de protection des fichiers sur la base de l'identificateur d'usager, fourni par le système d'exploitation Linux, sert donc de mécanisme pour assurer la séparation des fichiers entre les applications. Sur IOS, toutes les applications sont exécutées avec le même identificateur d'usager. Cependant, chaque application se fait attribuer un répertoire différent dont le nom long et aléatoire sert de clé d'accès. Une application, qui ne connaît pas le nom du répertoire d'une autre application, ne peut pas, en temps raisonnable, en deviner le nom.</p>	<p>b) Une machine virtuelle tourne sur un noeud physique Foo. L'image de cette machine virtuelle contient 8 000 000 pages. On veut migrer cette machine virtuelle vers un noeud Bar 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 5000 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 0.15 seconde. i) Combien de temps durera la migration au total? ii) Combien de temps est-ce que l'exécution sera arrêtée?</p> <p>Réponse : Lors du premier tour, les 8 000 000 pages seront copiées en $8\ 000\ 000 / 25\ 000 = 320s$. Pendant ces 320s, nous aurons $320s \times 5000p/s = 1\ 600\ 000$ pages modifiées. Au second tour, ces pages sont copiées en $1\ 600\ 000 / 25\ 000 = 64s$, temps pendant lequel $64s \times 5000p/s = 320\ 000$ pages sont modifiées. Au troisième tour, les pages sont copiées en $320\ 000 / 25\ 000 = 12.8s$, et $12.8s \times 5000p/s = 64\ 000$ pages sont modifiées. Au quatrième tour, les pages sont copiées en $64\ 000 / 25\ 000 = 2.56s$, et $2.56s \times 5000p/s = 12\ 800$ pages sont modifiées. Rendu là, l'exécution est interrompue, les pages restantes prenant moins que 0.15s à copier, et les 2560 pages restantes sont copiées en $2560 / 25\ 000 = 0.1024s$.</p> <p>i) Le temps total pour la migration est donc de $320s + 64s + 12.8s + 2.56s + 0.1024s = 399.9744s$.</p> <p>ii) L'exécution est arrêtée pendant le dernier tour qui dure <u>0.1024s</u>.</p> <p>iii) La fonction KSM (Kernel Same page Merging) dans le noyau du système d'exploitation Linux est un mécanisme qui est présenté comme pouvant améliorer l'efficacité des machines virtuelles. Que fait cette fonction? A quoi sert-elle? Pourquoi est-ce particulièrement utile en présence de machines virtuelles?</p> <p>Réponse : Avec KSM, une tâche en arrière-plan vérifie si deux pages ou plus, en lecture seulement, ont exactement le même contenu. Si c'est le cas, ces pages sont fusionnées tant qu'elles ne sont pas modifiées. Cela permet de réduire les besoins en mémoire. Ceci est particulièrement utile pour les machines virtuelles, puisqu'elles utilisent souvent le même contenu (même version d'applications et de bibliothèques dans chacune des machines virtuelles), par exemple)</p>	<p>Question 2 :</p> <p>a) Un système d'appel de procédure à distance utilise une sémantique au plus une fois avec le protocole sans connexion UDP. i) Expliquez combien de paquets et lesquels sont échangés entre le client et le serveur pour compléter un appel à distance, dans le cas simple où aucun paquet ne sera perdu, incluant ce qui est nécessaire pour libérer les ressources associées à cet appel dans le serveur. ii) Si le système d'appel à distance utilisait plutôt la sémantique au moins une fois, comment cela changerait-il?</p> <p>Réponse : i) Avec la sémantique au moins une fois, un paquet avec la requête, du client au serveur, et un paquet avec la réponse, du serveur au client, suffit. i) Pour la sémantique au plus une fois, le serveur mémorise le numéro de la requête et la réponse associée. Ainsi, si la requête est retransmise par le client, parce que le paquet de réponse a été perdu, la réponse mémorisée est retransmise. Cependant, avec cette sémantique, le client doit en plus envoyer un accusé de réception de la réponse, pour permettre au serveur de libérer l'espace pris par la réponse mémorisée. Cela fait donc 3 paquets au lieu de 2.</p> <p>b) Un appel de procédure à distance doit contenir les arguments suivants: string prenom, string nom, string candidat, int id. Si les valeurs sont: "Louis-Jean", "Symphorien", "Pikachu Ninjago", "778899", combien d'octets seront requis pour encoder cette information avec: i) CORBA CDR (32 bits)? ii) Avec gRPC protobuf?</p> <p>Réponse : Avec CORBACDR, nous savons 4 octets pour la longueur du prénom et 12 octets (multiple de 4) pour les 10 lettres de "Louis-Jean", 4 octets pour la longueur du nom et 12 octets (multiple de 4) pour les 10 lettres de "Symphorien", 4 octets pour la longueur du candidat et 16 octets (multiple de 4) pour les 15 lettres de "Pikachu Ninjago", et 4 octets pour le id. i) Le total est donc de $4+12+4+12+4+16+4 = 56$ octets. Pour le format protobuf, nous avons pour le prénom 1 octet type/champ, 1 octet longueur et 10 octets pour "Louis-Jean", pour le nom 1 octet type/champ, 1 octet longueur et 10 octets pour "Symphorien", pour le candidat 1 octet type/champ, 1 octet longueur et 15 octets pour "Pikachu Ninjago", et pour le id 1 octet type/champ et 3 octets pour la valeur 778899 (0b101111000101000101, soit 20 bits qui entrent dans 3 octets variant de 7 bits). ii) Le total est de $1+10+1+1+10+1+1+15+1+3 = 45$ octets.</p> <p>c) Le protocole SOAP (Simple Object Activation Protocol), basé sur XML, est souvent utilisé pour les appels de procédures à distance sur la Toile (Web). Quels sont les avantages et inconvénients de SOAP comparé à un protocole d'appel de procédure à distance comme gRPC avec protobuf?</p> <p>Réponse : Le protocole SOAP est entièrement textuel et facile à comprendre. Il est donc facile à programmer et à mettre au point. Cependant, il est beaucoup moins efficace et compact qu'un protocole binaire comme gRPC avec protobuf.</p> <p>Question 3 :</p> <p>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 <code>requete_client</code> est créé et une référence réseau à cet objet est retournée au client de la requête. Le client cesse d'utiliser cette référence réseau environ 80 secondes plus tard. Deux stratégies sont possibles pour prévenir le serveur lorsque l'objet de type <code>requete_client</code> n'est plus utilisé et peut être libéré. La stratégie notification procéde en envoyant un message pour se désenregistrer de cet objet réseau exporté. La seconde stratégie, bail renouvelable, précise une durée de vie par défaut de l'objet après laquelle l'objet est libéré, 120 secondes dans ce cas-ci, à moins qu'un renouvellement ne soit demandé. Si le processus serveur reçoit 1 requête par 10 secondes de chaque client, et sert 500 clients, combien d'objets de type <code>requete_client</code> sont présents simultanément dans le processus serveur en moyenne: i) avec la stratégie notification, ii) avec la stratégie bail renouvelable?</p> <p>Réponse : Le temps moyen qu'un objet reste alloué pour une requête avec la première stratégie est de 80s. Avec la seconde stratégie, ce temps est de 120s. Le taux d'arrivée des requêtes est de $1r/10s$ par client \times 500 clients = $50r/s$. i) Le nombre total d'objets alloués simultanément pour la première stratégie est donc de $80s \times 50r/s = 4000$. ii) Avec la seconde stratégie, le nombre total d'objets alloués simultanément est de $120s \times 50r/s = 6000$.</p>
---	---	--

<p>UDP, fast and less accurate, video streaming, VoIP call, recherche, look up</p> <p>TCP, slow but precise, web browsing, emails, files, bank</p> <p>RPC vs LPC, false or true</p> <p>a. "Les appels RPC nécessitent un paramètre supplémentaire pour identifier le serveur VRAI Le RPC doit savoir où envoyer l'appel (adresse du serveur distant)</p> <p>b. "Les appels RPC ont des erreurs différentes (plus nombreuses)": VRAI. dépend de l'infra réseau e. "Les appels RPC peuvent avoir une latence plus élevée ou variable" VRAI La latence réseau rend les appels RPC plus lents et moins prévisibles que les appels locaux</p> <p>c. "Les appels RPC sont limités à l'appel par valeur": FAUX Les RPC peuvent supporter différents types de passage de paramètres</p> <p>d. "Les appels RPC sont facturés comme des appels interurbains: F</p> <p>e. "Les appels de fonctions locales nécessitent un malloc d'un objet dans le tas pour représenter le pointeur de retour" FAUX Les appels locaux n'ont pas besoin d'allocation spéciale pour les retours</p> <p>SOAP : pas bon, Protocole de messagerie basé sur XML, obligatoire, gourmand, enveloppés, SMTP<</p> <p>REST: Style architectural pour systèmes hypermédia distribués (pas un protocole), anorexique, Format de message flexible (XML, JSON, etc.), only HTTP</p> <p>Les réponses correctes sont Les opérations à effectuer sont codées dans le document avec SOAP mais dans l'URL avec REST. SOAP doit utiliser XML pour coder ses données.</p> <p>Docker: Container platform that packages applications and dependencies together, LIGHT Creates standardized, isolated environments called containers, host OS kernel Great for development and testing environments</p> <p>Kubernetes: Container orchestration, Manages containerized applications across a cluster of machines, Key features Automated container deployment, autoscaling, load balancing Main components: Pods Groups of containers deployed together, Nodes Worker machines running containerized applications</p> <p>When to use which: - Docker: Single-host container deployments, development environments ---- Kubernetes: Production environments, complex applications requiring high availability and scaling</p> <p>NFS - Gestion des timeouts : Par défaut : accès bloquants En cas de timeout RPC NFS retourne le contrôle au processus avec un code d'erreur. Le serveur étant "stateless", le client peut réessayer plus tard Problème : certains programmes ne sont pas conçus pour gérer ces erreurs Points clés supplémentaires : Module client NFS dans le noyau Processus biod gère les lectures/écritures asynchrones Système "stateless" : le serveur peut redémarrer sans perdre l'état des connexions VFS Virtual File System) gère le routage vers les différents systèmes de fichiers Montage en "dur" des systèmes de fichiers Soit un système de fichiers NFS processus P essaie d'accéder à un fichier un appel RPC est émis. Si appel expire, action effectuée par le module client NFS? Réponses correctes : b. NFS retourne le contrôle au processus P c. NFS va refaire l'appel RPC</p> <p>CORBA CDR 32 bits Règles d'encodage Rectangle 4 octets, Caractère 1 octet String Header 4 octets, Longueur totale divisible par 4, Si non divisible → padding Entiers 4 octets par rectangle, Pas de header, Max 2^32, Un entier par rectangle Exemple String de 18 chars 20 octets 4 header 24 octets 2 octets padding car doit être divisible par 4 À retenir <input checked="" type="checkbox"/> Strings : (longueur de char de chaque string + padding) + header Entiers 4 octets, un par rectangle <input checked="" type="checkbox"/> Jamais 2 strings sur même rectangle exemple: message sont respectivement de 24, 399 et 106, et trois champs qui sont des entiers, 98, 115 et 21340. 24/4 0, good, 3991 // 4 0, good, 1062 // 4 0, good, 24 400 108 3 messages x 4 padding 3 entiers x 4 octets 556</p> <p>Services infonuagiques Le stockage d'instance est propre à chaque activation de l'instance. Son contenu est perdu lorsque l'instance est arrêtée. Le stockage de bloc (Block Storage) est comme un disque local. Son contenu persiste après l'arrêt de l'instance et il peut être accédé à nouveau par une nouvelle instance. Un stockage de bloc ne peut toutefois être attaché qu'à une seule instance à la fois, comme un disque local. Le stockage d'objets (Object Storage) est comme une page Web qui supporte GET et PUT. On peut lire son contenu complet, ou remplacer son contenu complet, mais il n'est pas possible d'en remplacer seulement une partie.</p> <p>Types de transparence dans les systèmes répartis : Accès : Opérations identiques sur ressources locales/distantes Localisation : Nom constant indépendant de l'emplacement Concurrency : Gestion automatique des accès simultanés Défectuosité : Continuité malgré pannes Performance : Maintien des performances avec la montée en charge Plateforme : Compatibilité multi-plateformes Réplication/Migration : Masquage des opérations techniques Réseau, serveur de disques SSD, combien de client on peut supporter Entrée 1 Gbps 1000 Mbps, Requis par client 20 Mbps Limite réseau 1000/20 = 50 clients</p>	<p>Nombre de clients maximal Nb de disques / temps de disque par client Taux d'arrivée: $\lambda = 100/\text{s}$, Temps de service: $\mu = 1/0.008\text{s} = 125/\text{s}$, Taux d'utilisation: $U = \lambda/\mu$ Le nombre moyen de requêtes $N = U/(1-U) = 0.8/0.2 = 4$ Temps d'attente moyen, $W = N/\lambda = 4/100 = 0,04 \text{ secondes}$ $W = N / 5 \times \lambda / 4 = 5 \times 100 / 0,008 = 0,008$ Temps écriture Temps lecture Temps validation Temps lecture 8x(0,80,30,4 (besoin lecture supp.) 0,2 (pas en cache)) x 15 ms x 30%</p> <p>Websocket : le protocole permet d'avoir des interactions asynchrones bidirectionnelles entre un client et un serveur. La technologie est basée sur les événements et fonctionne par-dessus TCP. Le protocole de routage BGP peut être utilisé pour effectuer des échanges d'informations sur le routage des paquets entre différents systèmes autonomes. C'est un protocole de routage dynamique JSON-RPC l'emploi du format JSON permet l'interopérabilité entre systèmes de différentes technologies alors que la technologie RPC permet d'y puiser les informations concernant les procédures devant être invoquées. Dans le cas d'une implémentation on pourrait l'utiliser en synergie avec le WebSocket pour assurer les transferts des données sérialisées avec JSON et d'invoquer à la destination les procédures requises après désérialisation. XML peut être utilisé pour l'échange de données entre systèmes différents en respectant un schéma prédéfini. Il peut être utilisé pour briser les dépendances entre technologies en servant de format de représentation des informations</p>
---	--

Un volume sur un service GlusterFS est configuré pour être Réparti, Répliqué (DistributedReplicate). Les serveurs S1, S2 et S3 offrent chacun une brique de disque de même dimension qui sont ainsi répliquées et forment le volume répliqué VRO. 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 VRO 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 50Mo/s, quelle serait la bande passante pour écrire des fichiers sur ce volume VRR? Pour lire des fichiers?

Les fichiers vont se répartir entre VRO et VR1. Ensuite, chaque fichier sur VRO ou VR1 sera réplié sur les 3 serveurs qui les composent. Ceci pourrait donner: S1 (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 $6 \times 50\text{Mo/s} = 300\text{Mo/s}$. 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 $2 \times 50\text{Mo/s} = 100\text{Mo/s}$.

Quelle(s) technique(s) lessystèmesdefichiersrépartiscommeGlusterFS et TCPUtilisentpour permettre une très grande mise à l'échelle? 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 repartir 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, où 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.

Une nouvelle version de l'image des poste d'une. Entreprise doit être copiée à partir d'un serveur vers chacun des 50 postes. Par souci de rapidité, le protocole UDP est utilisé en multidiffusion et chaque poste envoie un accusé de réception négatif pour chaque paquet manquant. Chaque paquet est numéroté et les stations redemandent les paquets manquants lorsque la séquence n'est pas complète. On néglige l'espace requis pour les en-têtes et la numérotation des paquets. L'image occupe 45×10^3 octets. La probabilité qu'un paquet n'arrive pas à un poste donné est proportionnelle à sa longueur $p = po \times n$, po désignant la probabilité par octet et vaut 10-7 et n est la longueur des paquets et doit être comprise entre 1 et 106. Si la longueur des paquets est de 106 octets, combien d'accusés de réception négatifs seront reçus suite à l'envoi initial de l'image, (avant la retransmission des paquets manquants qui pourraient à leur tour être perdus). De manière générale, quel est le problème d'avoir des paquets trop courts? Des paquets trop longs?

L'image sera donc découpée en 45×10^3 paquets UDP. La probabilité qu'un paquet soit perdu avant d'arriver à destination est de $p = 0.1$. Le nombre de paquets perdus pour un poste sera de $45 \times 10^3 \times 0.1$, mais puisque nous avons 50 postes, le nombre total d'accusés de réception négatifs sera de 225×10^3 . Avoir des paquets trop courts augmente significativement leur nombre, tandis que des paquets trop longs augmentent leur chance d'être perdus (d'après la loi de probabilité donnée).

Il vous est demandé de proposer, implémenter et déployer un système réparti pour une organisation qui désire mettre en place son système E-commerce et vendre ses produits en ligne. Les technologies et protocoles suivants vous sont proposés: Websocket, BGP, JSON-RPC, XML. Pour chacun des protocoles ou technologies cités, décrivez leur utilité dans le contexte d'application de votre solution.

On s'attend à ce que l'étudiant explique le rôle de chaque protocole dans l'application mise en œuvre.

- Websocket : le protocole permet d'avoir des interactions asynchrones bidirectionnelles entre un client et un serveur. La technologie est basée sur les événements et fonctionne par dessus TCP.
- Le protocole de routage BGP peut être utilisé pour effectuer des échanges d'informations sur le routage des paquets entre différents systèmes autonomes. C'est un protocole de routage dynamique
- JSON-RPC : l'emploi du format JSON permet l'interopérabilité entre systèmes de différentes technologies alors que la technologie RPC permet d'y puiser les informations concernant les procédures devant être invoquées. Dans le cas d'une implémentation on pourrait l'utiliser en synergie avec le Websocket pour assurer les transferts de données sérialisées avec JSON et d'invoquer à la destination les procédures requises après déserialisation.
- XML: peut être utilisé pour l'échange de données entre systèmes différents en respectant un schéma prédefini. Il peut être utilisé pour briser les dépendances entre technologies en servant de format de représentation des informations

Un système réparti dans un réseau local offre la possibilité de collaborer en partageant du code source à travers une application déployée à cet effet. Pour y avoir accès, les utilisateurs doivent être connectés au réseau par le biais d'un commutateur. Pourriez-vous décrire le processus entrepris par ce dernier (commutateur) après son démarrage pour permettre dans son fonctionnement de pouvoir filtrer et acheminer les messages vers chacun des utilisateurs (ordinateurs) connectés.

Lors du démarrage du commutateur, ce dernier n'a aucune information sur les manières de filtrer les trames qu'il devra recevoir. Il maintient une table d'adresses MAC pour stocker les correspondances entre les adresses physiques et les interfaces sur lesquelles elles sont connectées. Le commutateur va attendre l'arrivée de la première trame émanant d'un équipement directement connecté, enregistrer l'adresse MAC de ce dernier et le numéro de l'interface correspondante.

Il va ensuite effectuer une diffusion sur l'ensemble des interfaces actives exceptée celle d'origine. Seul l'équipement concerné par le message répondra et son adresse MAC source sera enregistrée de même que le numéro d'interface correspondant. Le processus se répète pour chaque équipement non présent dans la table d'adresse MAC jusqu'à ce que la convergence soit effective. Le commutateur peut alors acheminer les trames vers leurs destinataires en se basant sur l'information présente dans sa table d'adressage.

Les technologies de virtualisation suivantes vous sont proposées: Conteneur, VLAN, VPN, Machine virtuelle. Il vous est demandé de mettre en place une infrastructure qui devrait les emploier. Dans quelle mesure chacune d'elle pourrait être utilisée à bon escient pour rendre votre système fonctionnel? Les conteneurs pourraient être utilisés pour permettre de regrouper des applications et leurs dépendances dans un même environnement d'exécution. Cela faciliterait leur déploiement et orchestration. Aucune machine virtuelle n'est créée, seul l'OS hôte est utilisé pour leur exécution en se basant sur les espaces de noms. Si plusieurs sous-réseaux ou réseaux devraient être implémentés, une technologie comme le VLAN serait intéressante dans la mesure où elle permettrait de séparer logiquement différents domaines de diffusion grâce à un commutateur et éviter les coûts additifs d'installation d'équipements de couche 3 (routeur). Le VPN pourrait être déployé pour assurer un niveau de sécurité aux données qui vont transiter par le réseau public. Cette technologie permettra d'interconnecter des sites distants en créant un tunnel de communication de bout en bout entre les différents sites distants. Des en-têtes sont rajoutées aux paquets pour assurer que la communication inter-réseaux se fasse de manière transparente. Les différents sites n'ont pas connaissance de l'utilisation du réseau public. Pour rouler certaines technologies nécessitant un environnement particulier d'exécution, des machines virtuelles peuvent être créées et supporter l'OS requis à cet effet.

Dans le TP1, on vous demande de monter un dossier distant par le biais de l'option -v (pour -volume) lorsque l'on roule un conteneur avec docker run. Rappelez dans quelle mesure Docker peut isoler les conteneurs entre eux, et par quel moyen le montage peut être réalisé. Est-il nécessaire de passer par une interface réseau ?

Docker isole les conteneurs par un système d'espace de nom pour les processus et un système de fichier virtuel, qui les empêche par défaut de communiquer entre eux. Il est cependant possible de demander au système d'exploitation de faire un montage virtuel entre le conteneur et le système de fichier "réel" afin de partager des fichiers comme des données d'une application ou des fichiers de configuration.

Le TP2 met en situation un appel de procédure à distance avec gRPC.

Expliquez le rôle de chaque fichier operation.proto, manager.cc et server.cc pour cette application

- operation.proto correspond au fichier d'IDL (Interface Description Language) qui régit les types de messages échangés (OperationRequest et OperationReply par exemple dans notre cas) ainsi que les différents services d'appel de procédure à distance.

- manager.cc incarne un client dans le cadre de l'appel de procédure à distance. gRPC permet de ne faire appel qu'à un stub de fonction qui réalise lui-même l'appel de procédure à distance.

- server.cc est le fichier où est implémenté la fonction définie. C'est un serveur qui sert les appels de procédure à distance, qui sont implémentés à partir d'une classe abstraite squelette créée par gRPC.

La recherche d'un objet dans un système réparti peut s'effectuer par envoi à tous. Pourriez-vous expliquer comment se déroule ce processus dans le cas de la recherche d'une correspondance dans un réseau communiqué (commutation de circuits)? Qu'entendez-vous par filtrage de trames au niveau du commutateur? Dans ce cas, une table de routage est-elle nécessaire pour acheminer les paquets à leur destination?

Dans le cas du commutateur, la recherche d'un objet par envoi à tous est employée lorsqu'un équipement connecté dans un réseau ethernet grâce à un commutateur ne possède l'adresse IP mais pas l'adresse MAC de l'équipement de destination. Une requête ARP (Adresse Resolution Protocol) est alors envoyée à tous. Cette dernière est diffusée par le commutateur aux différents postes connectés et seul le poste concerné renvoie la réponse à l'expéditeur. La correspondance MAC étant trouvée, le paquet peut être alors constitué avec l'ensemble des ses en-têtes et envoyé vers sa destination. L'une des opérations que le commutateur effectue est le filtrage des trames. Il est dans ce cas capable de sélectionner l'interface sur laquelle est connectée le destinataire d'un message. Une table de routage n'est pas nécessaire, car seul un segment réseau est utilisé et il n'est pas besoin d'acheminer des paquets vers un réseau différent. Le commutateur achemine les messages en se basant sur les adresses ethernet des différents postes.

Un service de fichiers CODA est répliqué sur 3 serveurs (i.e. Chaque fichier se retrouve en 3 copies). Chaque serveur possède 4 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 3 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)? Si on change pour un système avec 3 serveurs mais sans réPLICATION, avec la charge uniformément répartie entre les 3, que devient le nombre maximal de lectures? Le nombre maximal d'écritures?

En lecture, le système de fichier peut satisfaire $3 \times 4 \times 100 = 1200$ requêtes/s. En écriture, en revanche, il faut répliquer l'accès sur les trois serveurs ce qui donne $4 \times 100 = 400$ requêtes/s.

Si le système de fichiers est distribué, les trois serveurs fonctionnent toujours en parallèle pour la fois lectures et écritures. On obtient que le système peut supporter $3 \times 4 \times 100 = 1200$ requêtes/s.

Un serveur Oracle reçoit en moyenne 100 requêtes par seconde, arrivant selon un processus de Poisson. Les requêtes reçues par le serveur sont stockées en queue dans un disque de grande capacité, lorsque ce dernier est occupé, pour être traitées ultérieurement. Deux types de serveurs peuvent être utilisés en fonction de leur capacité pouvant respectivement traiter 60 et 120 requêtes par seconde en fonctionnant sans interruption. Qu'arrive-t-il si l'on utilise le premier serveur? Le deuxième? Quel est le temps de réponse moyen dans chacun des cas?

Choisis le premier serveur mènerait à un taux d'utilisation $U \geq 1$, qui implique une file d'attente infinie (et donc un temps de réponse infini aussi) et disqualifie donc ce serveur. Pour le second, on obtient $U = 100 / 0.833 = 120 \mu\text{s}$. Le temps de réponse moyen est $W = 1 / 0.05 = 0.05 \text{ s}$.

Un serveur de disque composé de disques SSD traite les requêtes de clients. 20 mégabits/s en moyenne sont requis par chaque client dans un réseau communiqué. Le serveur est connecté au réseau par une prise fournissant 1 gigabit/s. Son bus a une capacité de 8 gigaoctets/s avec 6 disques connectés. Ces derniers fournissent 50 mégaoctets/s. Les chiffres fournis sont des puissances de 10 (e.g. mega = 106 et giga = 109). Combien de clients peut-il supporter?

Le réseau peut fournir 1 Gb/s, le bus 8x8 = 64 Gb/s et les disques 0.050x8x8 = 2.4 Gb/s. Le plus lent est le réseau, qui peut servir jusque 1 = 50 clients.

Le système Android est basé sur le système d'exploitation Linux. Sur Linux, les permissions sur les fichiers sont généralement exprimées sous la forme de permissions données à l'utilisateur, au groupe ou aux autres pour: la lecture, l'écriture, et l'exécution. Toutefois, pour déterminer quelles permissions sont données à chaque application Android, une granularité plus fine est souvent requise. Par exemple, différentes informations, qui demandent des permissions différentes, pourraient se retrouver dans le même fichier. Une permission peut aussi être donnée temporairement, au moment où l'application est exécutée, après validation auprès de l'utilisateur. Comment cela est-il réalisé sur Android pour donner ou non les permissions aux applications, avec une granularité plus fine que celle permise pour les fichiers par Linux?

Android offre de nombreux services qui sont accessibles via des daemon. Ces daemon sont des processus serveur qui roulent en arrière-plan et acceptent des messages inter-processus de manière temporaire. Ainsi, la validation des permissions se fait dans ces processus, après avoir validé l'identité du processus ayant envoyé la requête, et peut donc être beaucoup plus flexible et granulaire que ce qui est offert normalement pour les fichiers dans le système d'exploitation Linux.

Vous êtes l'expert en informatique et des collègues viennent vous consulter pour savoir s'ils devraient utiliser des machines virtuelles (e.g. avec KVM ou VirtualBox) ou des conteneurs (e.g. Kubernetes et Docker qui utilisent les cgroup sur Linux). Le premier collègue veut rouler sur le même système embarqué, qui roule Linux, de nouveaux services sur Linux, ainsi qu'un ancien service fourni par une ancienne application Windows. Le second collègue doit bâti un serveur qui compile une application pour plusieurs versions différentes de distributions Linux (e.g. Ubuntu 20.04, Ubuntu 22.04, Fedora 35, Fedora 36). Pour chaque version de distribution, il faut utiliser la bonne version du compilateur et les bonnes versions de bibliothèques, mais la version du noyau du système d'exploitation Linux importe peu. Que suggérez-vous d'utiliser, machine virtuelle ou conteneur, dans chaque cas? Pourquoi?

Pour le premier cas, pour exécuter une application Windows, il faudra prendre une machine virtuelle afin de pouvoir exécuter ce système d'exploitation différent. Il serait aussi possible d'envisager un émulateur d'API comme Wine, mais cette solution ne fait pas partie du choix proposé. Pour le second cas, des conteneurs feront très bien l'affaire. Chaque conteneur peut venir avec une distribution différente (bibliothèques, compilateurs...) mais ils sont contraints à utiliser le noyau Linux de l'hôte, ce qui est spécifiquement n'étant pas un problème. Lorsque c'est possible, on préfère utiliser des conteneurs, plutôt que des machines virtuelles, car le surcroît est beaucoup moindre et le temps de démarrage est beaucoup plus court.

Un processus serveur reçoit des requêtes de clients par le biais d'appels de méthode à distance. Le serveur reçoit 25 requêtes par seconde et chaque requête crée un nouvel objet réseau de type session qui sera utilisé pendant 350 secondes. On envisage deux stratégies possibles pour déterminer quand les objets réseau peuvent être libérés. Pour la première stratégie, une notification est envoyée par le client lorsque l'objet n'est plus utilisé. Cependant, on estime que pour 1% des requêtes, le message de notification ne parviendra pas au serveur et ainsi l'objet ne sera pas libéré et restera en mémoire dans le serveur. Pour cette raison, le serveur est redémarré au milieu de chaque nuit afin de repartir à 0 et que les objets ne s'accumulent pas d'un jour à l'autre. Pour la seconde stratégie, l'objet est créé pour une durée de bail de 500 secondes, durée qui peut être prolongée au besoin en demandant une extension de bail de 500 secondes à la fois. Quel est le nombre d'objets réseau de session qui se retrouvent simultanément en mémoire dans le serveur dans le pire cas pour la première stratégie? Pour la seconde?

Avec la première stratégie, nous avons 1% des requêtes qui créeront un objet qui ne sera pas libéré avant la fin de la journée. Ceci crée une accumulation de $0.01 \times 25/\text{s} \times 60/\text{s} \times 60\text{m/h} \times 24\text{h/j} = 21600\text{r/j}$, soit 21600 objets orphelins à la fin de la journée avant le redémarrage. En plus, il y a les objets actifs. Lorsqu'une première requête arrive, elle ne sortira qu'après 350s, le nombre de requêtes présentes simultanément (entrées avant que la première ne sorte) sera donc de $25/\text{s} \times 350\text{s} = 8750$, soit autant d'objets réseau. Le total avant de redémarrer est donc de $21600 + 8750 = 30350$. Avec la seconde solution, nous aurons $25/\text{s} \times 500\text{s} = 12500$ objets. Il y a un peu plus d'objets (que ceux actifs dans la première stratégie) avec la seconde stratégie, car ils sont conservés pour 500s, alors qu'ils auraient pu être libérés au bout de 350s. Par contre, on sauve au niveau des messages de notification qui ne sont plus requis, et surtout en évitant les fuites causées par les notifications manquantes.

Le travail pratique 3 a mis en contexte un scénario dans lequel vous disposez de plusieurs serveurs formant une grappe de calcul. Vous avez dû implémenter un gestionnaire permettant de soumettre des tâches à exécuter selon une file d'attente FIFO. Pour cela, vous avez utilisé le système d'appel de procédure à distance gRPC. Deux fichiers principaux devaient être modifiés lors de cet exercice: "Manager.cc" et "operation.proto". Expliquez le rôle que joue chacun de ces fichiers et détaillez la démarche suivie pour définir les éléments nécessaires pour rendre votre implémentation fonctionnelle.

Le fichier Manager.cc avait pour rôle de prendre les opérations, de les mettre dans une queue FIFO et de les distribuer aux différents serveurs. Le fichier Operation.proto avait pour rôle de définir les messages et les services selon le langage de description d'interface (IDL) de gRPC. ProtoC utilise ces définitions pour générer les classes utilisées par Manager.cc (proxy) et Server.cc (squelette). Dans le travail pratique, il n'y avait qu'une seule opération donnée en argument et un seul serveur. Tout d'abord, il a fallu définir les messages et les services selon l'implémentation du service dans server.cc et manager.cc. Ensuite, il a fallu passer l'argument le programme principal (main) à la méthode ProcessOperation. Finalement, il a fallu rajouter l'appel gRPC vers le serveur.