INF4410

Module 3 : Systèmes de fichiers

3.1 Dans les premiers systèmes poste-à-poste comme Napster, la mise à l'échelle souffrait de l'existence d'un index centralisé. Proposez d'autres solutions d'indexation qui peuvent se mettre à l'échelle.

L'infrastructure d'indexation peut être mise à l'échelle en divisant l'index entre un grand nombre d'ordinateurs. On peut aussi miser sur l'indexation par Google qui fonctionne de cette manière. Il est aussi possible de répliquer l'information d'index entre de nombreux nœuds qui participent au réseau poste-à-poste en supposant que l'index n'est pas trop grand et que les mises à jour ne sont pas trop fréquentes. Par exemple, les nœuds adjacents peuvent se partager l'index pour éviter d'avoir chacun à le stocker et maintenir au complet.

3.2 Les réseaux poste-à-poste dépendent en partie de la bonne collaboration de plusieurs utilisateurs/serveurs. Quelles peuvent être les conséquences de participants égoïstes ou malicieux, donnez des exemples?

L'utilisateur idéal accepte de servir en échange de se faire servir et offre du contenu valide. Un utilisateur égocentriste peut ne vouloir que recevoir et ne rien envoyer. La plupart des systèmes ajustent la priorité d'un client pour recevoir en fonction de ce qu'il a envoyé. Il serait possible de fausser ces chiffres. Un client malicieux pourrait vouloir nuire aux recherches de fichiers en offrant des fichiers semblables mais au contenu vide, corrompu, ou possiblement avec un virus. Dans certains cas, un client malicieux pourrait aussi offrir des fichiers protégés par les droits d'auteur et noter les clients qui en prennent une copie.

3.3 Certains réseaux poste-à-poste offrent un service qui veut garantir l'anonymat. Est-ce possible?

Si plusieurs nœuds intermédiaires sont utilisés pour établir une connexion et que les nœuds intermédiaires ne divulguent pas leur information, ceci est possible en théorie. Si un bon nombre de ces nœuds intermédiaires sont compromis (e.g. un sur deux), il peut être possible de déduire l'information manquante. De plus, si celui qui veut percer l'anonymat a accès au réseau en plusieurs points, il peut suivre le cheminement des paquets et même en envoyer lui-même comme sonde. Une technique pour empêcher en partie cela serait d'envoyer chaque paquet reçu à plus d'une adresse, de sorte qu'il serait plus difficile de distinguer le vrai destinataire en épiant le cheminement des paquets.

3.4 Vous désirez rendre disponible un fichier de 1GiO à 100 clients. Chaque client a une bande passante de 1MiO/s dans les deux directions. On vous propose deux méthodes, un serveur de type Napster ou un autre de type BitTorrent. Supposez que vous avez un serveur (Napster ou BitTorrent) et initialement un client

contenant le fichier à distribuer (votre ordinateur). Les échanges de contrôle avec les serveurs et même entre les clients sont négligeables. Combien de temps cela prendrait-il environ avec chacune des méthodes, si le transfert est fait efficacement, sachant que BitTorrent fonctionnera avec des morceaux de fichiers d'une longueur de 1MiO.

La copie à 1MiO/s prendra 1000s. Avec Napster, dans le meilleur des cas, il faut y aller par étapes: 1 client prend une copie, ensuite 2 clients, puis 4, 8, 16, 32, 64 (total 127). Ceci prend donc 7 tours ou 7000s. Avec BitTorrent, le même scénario peut se faire morceau par morceau. L'envoi d'un morceau prend 1s. Après 7s, le premier morceau a rejoint tout le monde, pendant ce temps, d'autres morceaux sont envoyés à d'autres clients. Après 100s, chaque client pourrait avoir reçu un morceau différent et tous les clients peuvent en parallèle envoyer et recevoir un morceau. A ce rythme, les 1000 morceaux peuvent se propager aux 100 clients en 1000s. Toutefois, il faut néanmoins 7s après la réception du dernier morceau pour qu'il se propage. Ainsi, avec ce scénario approximatif, on peut voir comment la propagation se ferait en 100+1000+7= 1107s environ.

3.5 Pourquoi n'y a-t-il pas de open/close dans le service de fichiers réparti présenté en exemple?

Ces fonctions impliquent un état conservé dans le serveur, ce qu'un service réparti veut éviter pour continuer à opérer de manière transparente même si le serveur se réinitialise intempestivement.

3.6 Pourquoi avoir des identificateurs de fichiers uniques à travers le réseau? Comment les produire?

Cela permet de faire référence à un fichier sans avoir à y ajouter un contexte (serveur). Un tel identificateur unique peut être obtenu avec le numéro IP de la machine de création, concaténé à un numéro unique (compteur séquentiel) sur cet ordinateur.

3.7 En quoi le comportement local peut-il différer du comportement avec NFS pour l'accès concurrent à un fichier (POSIX copie unique)?

Lorsqu'un programme modifie un fichier en local, les autres programmes voient immédiatement la modification. Par NFS, un programme sur un autre client pourrait utiliser une ancienne version en cache jusqu'à 3 secondes après l'écriture du premier programme.

3.8 Quelle information le module client NFS doit-il conserver pour les processus qui l'utilisent?

Pour chaque fichier ouvert, le client NFS doit conserver le vfs inode, la position courante dans le fichier, et l'identificateur NFS du fichier.

3.9 Quelle faille de sécurité était présente avant NFS 3, est-elle complètement corrigée?

Au départ le client spécifiait en argument son identité et pouvait ainsi raconter n'importe quoi au serveur. Avec la version 3, une clé DES est obtenue au moment du mount. Il serait possible de refiler de fausses informations au moment du mount de manière à obtenir frauduleusement la
clé ainsi.

3.10 Quelle est l'utilité de monter en dur par NFS?

Plusieurs programmes ne vérifient pas les erreurs sur les opérations comme read/write. Un éditeur de texte pourrait essayer de sauver un fichier et présumer avoir réussi alors que ce n'est pas le cas. Dans un tel contexte, il est mieux que l'opération bloque plutôt que de retourner un code d'erreur qui serait ignoré.

3.11 Comment le serveur automount permet-il d'augmenter la fiabilité et la performance de NFS?

Il est possible de spécifier plusieurs serveurs pour un sous-arbre sous la responsabilité de automount. Il prendra le serveur qui répond le plus rapidement, ce qui contourne les serveurs défectueux ou trop lents. Ceci se limite normalement aux sections de l'arbre qui ne changent pas, sont accédées en lecture seulement, et sont donc répliquées facilement.

3.12 Comment AFS tient-il compte des messages de modifications qui seraient perdus?

Lors d'une ouverture de fichier, si aucun message n'est arrivé du serveur depuis quelques minutes, le client se réenregistre pour recevoir les messages de modifications. Il ne sert à rien de revérifier alors que le fichier n'est pas utilisé.

3.13 Pourquoi AFS peut-il supporter plus de clients que NFS?

Avec AFS, les messages de modifications évitent les très nombreuses demandes de revalidation de NFS. De plus, l'ouverture d'un fichier fait en même temps sa lecture du serveur, et la fermeture fait au besoin son écriture sur le serveur. Cela réduit le nombre total d'appels au serveur.