Parcours DataScale
Architecture des Gestionnaires de Données

Examen 24 février 2016

Durée : 2h30. Tous (vos) documents autorisés.
Soyez précis dans vos réponses, toute réponse non justifiée sera considérée comme fausse.
Utilisez deux feuilles séparées pour les Parties 1 et 2.

PARTIE 1 (13 points) : Mécanismes internes des Gestionnaires de Données

Le site MyGifts crée une table `Livraison(NumLiv, NbColis, Type, Poids, CodePostal)` pour organiser les livraisons de cadeaux de Noël. La table contient 1 million de tuples et est hachée sur `Type` (8 valeurs possibles) et `CodePostal` (36000 valeurs possibles) par hachage multi-attributs. Le numéro de paquet de hachage est composé de p1 bits résultant de l'application d'une fonction de hachage h1 sur `Type` et de p2 bits résultant de l'application d'une fonction de hachage h2 sur `CodePostal`.

Q1.1 (6 points) : calculer en fonction de p1 et p2 le nombre d'accès disques (i.e., nombre de paquets à lire) nécessaires pour exécuter les requêtes suivantes:

(a) Recherche des Livraisons tels que `Type` = 'Express'

(b) Recherche des Livraisons tels que `Type` = ' Express' et `CodePostal` = 78000

(c) Recherche des Livraisons tels que `Type` = ' Express' ou `CodePostal` = 78000

(d) L'ordre dans lequel on range les bits p1 et p2 pour former la signature d'un tuple (i.e., le résultat de la fonction de hachage appliquée à ce tuple) a-t-il une importance : (1) dans le cas d'un hachage statique (nombre de paquets invariant fixé dès la création du fichier), (2) dans le cas d'un hachage extensible ?

(e) Une alternative aurait été de créer un index bitmap sur `Type`. Quelle aurait été la taille de cet index bitmap en nombre de pages disque (on suppose des pages de 4Ko).

(f) Un index secondaire de type B-Tree aurait-il permis d'accélérer des requêtes non accélérables par un bitmap ou du hachage: (1) sur l'attribut `Type`, (2) sur l'attribut `CodePostal` ?

Q1.2 (4 points) : Trois types de transactions s'exécutent en parallèle sur ce site. Les transactions de type T1 ne font que des insertions dans la table `Livraison` (une transaction = une seule insertion), celles de type T2 calculent des statistiques globales (ex: quantités moyennes de produits livrés par CodePostal) et celles de type T3 font des recherches pour déterminer de façon stable (i.e., isolée des effets des autres transactions) la liste des produits à livrer dans une ville.

(a) Quel est le degré d'isolation le mieux adapté à chaque type de transaction ?

(b) Précisez quelles situations de blocage peuvent se produire entre ces transactions (on fait l'hypothèse d'un verrouillage de granule page). Dit autrement, quels types de transaction Ti peuvent se bloquer en attente de quels types de transactions Tj, avec i=j ou i≠j ?

(c) Une situation de verrou mortel est-elle possible ? Justifiez votre réponse.

(d) Un verrouillage hiérarchique tuple/page (i.e., par intention) permettrait-il d'accélérer les traitements et/ou réduire les conflits ? Si oui, de quelle façon (quel type de transaction poserait quel type de verrou) ?

Q1.3 (3 points) : Quiz (Attention, toute réponse non justifiée sera considérée comme fausse)

(a) Dans le modèle d'indexation de Cassandra, on produit un ensemble de SSTable (index partiels) que l'on fusionne à une certaine fréquence et sur lesquels on défini des Bloom filters. Si l'on augmente la fréquence de fusion des SSTable, les Bloom filters sont-ils toujours utiles ?

(b) Dans un protocole de validation atomique à 2 phases (2PC), existe-t-il des cas où une transaction est validée alors qu'un des participants tombe en panne ? Si oui, à quel moment doit se situer cette panne ? Sinon, pourquoi ?

(c) Les In-Memory Database Systems (IMDS) le modèle de stockage est massivement basé sur des pointeurs. Pourquoi un tel modèle n'est-il pas utilisé dans les SGBD traditionnels (orientés disque) ?

PARTIE 2 (7 points) : Un serveur d'images dans le cloud

Dans cet exercice, vous êtes en charge pour votre entreprise du développement d'une application web de stockage d'images de type Instagram, Flickr ou DevianArt. La conception initiale est en phase avec les bonnes pratiques de création d'une application web orientée cloud and fonctionne comme suit. Nous distinguons la couche d'application et la couche de stockage. La couche d'application exécute la logique métier alors que la couche de stockage assure la persistance des données. Chaque serveur d'application est connecté à un serveur de stockage et accède en lecture et écriture aux données selon la logique métier.

Le concept clé de l'application est une classe Album qui fournit deux opérations: addImage(Image i):void et readImages():List. L'opération addImage(Image i) ajoute l'image i à l'album alors que readImages() retourne la liste des images contenues dans l'album. Dans la couche de stockage, la classe Album persiste en utilisant Infinispan (ou un serveur de données clé-valeur analogue). Le système de stockage offre une abstraction de Cache au développeur lui permettant d'écrire (put) et lire (get) une donnée à partir d'une clé. Cette abstraction est similaire au concept de Map communément répandu dans les langages orienté objet comme Java ou C++. Chaque image est stockée en utilisant le hash de son contenu comme clé. Nous stockons la collection d'images via le hash du nom de son album. Le pseudo-code est donné ci dessous :

```
class Album{
        Cache cache
        String name
        void addImage(Image i) {
                // TODO
        }
        List readImages() {
                List images = new List()
                List keys = cache.get(hash(name))
                for (String k : keys) {
                        images.add(cache.get(k))}
                return images
        }
}
```

Q2.1. Pour le moment, nous considérons le cas d'un serveur d'application mono-threadé. A partir de l'exemple de *readImages*(), écrivez le pseudo-code de l'opération *addImage(Image i).*

Q2.2. Avec l'aide du service marketing de la compagnie, nous avons estimé le pic de charge de l'application à 200 000 opérations/seconde. Des mesures ont montré que (i) une application mono-threadé est capable de supporter au plus 1 000 opérations/seconde, et (ii) chaque serveur de la compagnie peut supporter au plus 16 threads simultanément. A la mesure de ces chiffres, pensez vous que l'application doit monter en charge horizontalement, verticalement ou les deux pour supporter le pic de charge?

Q2.3. Pour une charge basse/pleine, un serveur consomme 10/22 euros par heure. La pleine charge est atteinte à partir de 8 threads actifs. Déterminer le nombre de serveurs requis pour exécuter le pic de charge au coût le plus faible.

Q2.4. Nous introduisons maintenant quelques modifications dans notre conception pour assurer la scalabilité de notre application. Quel est le mot clé Java à utiliser pour assurer qu'un thread à la fois puisse lire/écrire une image dans l'album?

Q2.5. Nous considérons que la classe Album a été modifiée pour protéger des threads concurrents lors des accès. De plus, nous supposons que des serveurs d'applications multiples peuvent s'exécuter concurremment et que les opérations sur Cache sont linéarisables. Qu'est ce qui arrive si deux serveurs d'applications ajoutent de manière concurrente une image dans un album? Proposez une approche simple pour éviter ce problème.

Q2.6. Les personnes du service communication de la société sont intéressés par avoir quelques mesures statistiques sur les albums photos. En particulier ils veulent connaître la distribution des extensions des fichiers images (e.g 40% de png et 12% de jpg) pour proposer le meilleur outil de visualisation d'images aux utilisateurs. Expliquer de manière globale comment utiliser le modèle map-reduce pour implémenter cette fonctionnalité.