XMS1IE071 - Données Massives et Cloud

Pascal Molli - Jeudi 12 décembre 2024 (2h)

Les documents de cours sont autorisés (et seulement ceux-là.).

1 Exercice

Les questions avec des A, B, ..., Z sont à choix multiples. Répondez sur la feuille.

1.1 Question

Quelle caractéristique décrit le mieux une architecture sans état (stateless) ?

(A) Les informations de session sont stockées côté serveur entre les requêtes.

(B) Chaque requête du client contient toute l'information nécessaire pour comprendre et traiter la requête.

(C) L'état de la session est stocké dans une base de données persistante.

(D) Les requêtes du client doivent être synchronisées pour maintenir la consistance de l'état.

1.2 Question

Question : Pourquoi est-il recommandé d'écrire une API sans état (stateless) pour une application CLOUD ?

(A) Pour permettre une meilleure scalabilité de l'application.

(B) Pour augmenter la dépendance entre les requêtes et améliorer le suivi des utilisateurs.

(C) Pour réduire la complexité de la gestion des données utilisateur.

(D) Pour faciliter la sauvegarde des données sur le serveur.

1.3 Subsection

Quelle affirmation décrit le mieux la différence entre Platform as a Service (PaaS) et Infrastructure as a Service (IaaS) ?

(A) PaaS fournit des machines virtuelles et des réseaux, tandis que IaaS offre des plateformes de développement complètes.

(B) PaaS inclut la gestion des systèmes d'exploitation, des serveurs, du stockage et des réseaux, tandis que IaaS fournit uniquement des environnements de développement et des frameworks.

(C) IaaS fournit des machines virtuelles, du stockage et des réseaux, tandis que PaaS offre un environnement de développement complet avec des outils et des services gérés.

(D) IaaS et PaaS offrent tous deux des environnements de développement et de déploiement d'applications, mais IaaS est plus flexible en termes de personnalisation des outils.

1.4 Question

Qu'y a-t-il dans une offre PaaS ?

(A) Infrastructure physique (serveurs, stockage, réseaux)

(B) Environnements de développement, bases de données, et gestion de middleware

(C) Applications prêtes à l'emploi pour les utilisateurs finaux

(D) Services de support technique 24/7

1.5 Question

Google app engine fait partie de l'offre :

(A) IaaS

(B) SaaS

(C) LaaS

(D) PaaS

(E) ZaaS

1.6 Question

Quelle est la meilleure description du modèle de computing serverless dans le cloud ?

(A) Une architecture où les utilisateurs doivent gérer et scaler leurs serveurs selon les besoins.

(B) Un modèle où les serveurs sont totalement absents, permettant une optimisation maximale des ressources.

(C) Un paradigme où le fournisseur de cloud gère la scalabilité et l'allocation des ressources serveur, rendant ces derniers invisibles aux développeurs.

(D) Une stratégie de déploiement où toutes les applications doivent être écrites sans utiliser de serveurs.

1.7 Question

Quel est le principal objectif du "salage de clé" (key salting) lors de la conception de clés pour les entités dans un datastore distribué ?

(A) Améliorer la sécurité des données en chiffrant les clés

(B) Augmenter la vitesse d'accès en créant des indices supplémentaires

(C) Éviter les conflits de clés en ajoutant des préfixes

(D) Réduire les hotspots en répartissant les requêtes de manière uniforme

1.8 Question

Quelle est la différence principale entre le partitionnement et le sharding ?

(A) Le partitionnement divise les données entre plusieurs instances de bases de données, tandis que le sharding divise les données au sein d'une seule instance de base de données.

(B) Le partitionnement est utilisé pour améliorer la sécurité des données, tandis que le sharding est utilisé pour améliorer la performance des requêtes.

(C) Le partitionnement divise les données au sein d'une seule instance de base de données, tandis que le sharding divise les données entre plusieurs instances de bases de données.

(D) Le partitionnement est une technique de compression des données, tandis que le sharding est une technique de duplication des données.

1.9 Question

Quels sont les services offerts par le cloud provider pour google app engine ? :

(A) Le datastore

(B) Le cache

(C) Les transactions

(D) L'authentification

(E) La sécurité réseau

(F) L'élalisticité

1.10 Question

Sur Google app engine, des servlets "stateful" passent à l'échelle ?

— Vrai

— Faux

1.11 Question

L'elasticité permet le passage à l'échelle de n'importe quelle application ? :

— vrai

— faux

1.12 Question

Si 4/5 du code de mon programme peut-être parallélisé et que je dispose de 2 processeurs, quelle est l'accélération maximale atteignable ?

1.13 Question

Si 1/5 du code est séquentiel. Quelle est le nombre de processeurs nécessaire pour multiplier la vitesse par 2 ?

1.14 Question

Si 1/5 du code est séquentiel. Combien de processeurs sont nécessaire pour espérer multiplier la vitesse d'exécution du programme par 5 ?

1.15 Question

Le sharding du Google Datastore répartit uniformément

(A) les données sur les noeuds de données

(B) les accès aux données sur les noeuds de données,

1.16 Question

Quel est le principe du range partitioning dans les bases de données distribuées ?

(A) Distribuer les données de manière uniforme à travers toutes les partitions en utilisant une fonction de hachage.

(B) Diviser les données en segments basés sur des plages de valeurs spécifiques pour chaque partition.

(C) Compresser les données pour économiser de l'espace de stockage.

(D) Répliquer les données sur plusieurs partitions pour améliorer la tolérance aux pannes.

1.17 Question

J'ai fait une classe Person définie comme :

```
1 Person:
2   name: String
3   friends: ListOf(String)
```

Quelles sont les requêtes que je dois absolument exécuter avec un index composite :

(A) select * from Person WHERE name>'bob' AND name<"yoda"

(B) select * from Person WHERE friends='bob' AND friends = 'James'

(C) select * from Person WHERE name>'bob' AND name<"yoda" order by DESC(name)

(D) select * from Person WHERE name>'bob' AND friends = 'James'

(E) select * from Person WHERE name='bob' AND friends = 'James' order by name

(F) select * from Person WHERE name='bob' AND friends > 'James' order by name, friends

(G) select * from Person WHERE name>'bob' AND name<"yoda" order by DESC(name),ASC(friends)

1.18 Question

Pourquoi Google Datastore ne permet-il pas les jointures ?

(A) Parce que les jointures ne sont pas nécessaires dans les bases de données NoSQL.

(B) Parce que les jointures complexifient le schéma des bases de données.

(C) Parce que les jointures peuvent nuire aux performances et à la scalabilité dans une base de données distribuée.

(D) Parce que Google Datastore utilise un langage de requête différent du SQL.

(E) Parce que le modèle de prix en serverless fait payer les résultats intermédiaires.

1.19 Question

Par défaut, Le datastore google assure :

(A) la cohérence forte pour tout type de transaction

(B) l'atomicité de l'écriture d'un tuple

(C) l'atomicité pour des requêtes portant sur plusieurs tuples.

1.20 Question

Le datastore Google permet :

(A) d'exécuter des joins entre entités de plusieurs classes

(B) d'exécuter des joins entre entités de la même classe

(C) d'avoir des attributs "listes"

(D) de requêter des attributs "listes"

1.21 Question

Quelles sont les quatre propriétés garanties par une transaction ?

1.22 Question

Si T1 et T2 sont 2 transactions. On dit que la cohérence est forte si l'exécution en parallèle de T1 et T2 est équivalente (même read, même write) à l'exécution de T1 suivie de l'exécution de T2.

— Vrai

— Faux

2 TinyLog

On considère une application web gérant les journaux d'un serveur web. Une entrée dans le journal est constituée de :

— Une addresse IP

— un User ID

— Un horodatage

— The URL requested

— The HTTP status code

— La taille de l'objet retourné

Par example :

```
127.0.0.1 frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif
HTTP/1.0" 200 2326
...
```

L'application en question a stocké les entrées du journal dans le datastore en utilisant comme clé l'horodatage des entrées. Chaque fois que le serveur web reçoit une requête, nouvelle entrée est écrite dans le datastore.

Malheureusement, une fois en exploitation, les performances en écriture sont mauvaises.

Questions :

1. Que se passe-t-il ? ? Quel est le nom du phénomène observé ? ?
2. Proposer une méthode résoudre ce problème
   (a) écrivez le pseudo-code de l'opération écrivant une entrée du journal dans la datastore
   (b) Montrez un example d'une entrée dans le datastore.
3. écrivez la requête permettant d'afficher les entrées du journal à un jour donné et une heure donnée.

3 Exercice

Supposons une classe Animal definie comme :

```
class Animal:
 has = StringListProperty()
 color = StringListProperty
 legs = IntegerProperty
```

Les entités (bizarres :) de cette classe sont :

key	has	color	legs
antelope	tail	yellow	4
bear	jaws	brown	2
bison	tail	yellow	4
cat	tail	brown	4
cow	horns	brown	4
dog	tail	yellow	4
elephant	tail	yellow	5
lion	horns	brown	4

Je veux exécuter la requête suivante Q_1 :

```
SELECT * FROM Animal WHERE
  color='brown' AND has='horns' AND legs='4'
```

1. Construisez les index de ces entités, 1 index par propriété.
2. Combien d'accès sont nécessaires pour obtenir un résultat complet ? Numérotez vos appels sur vos index.

Je pose un index composite sur 'has' et 'color', et un index sur 'legs'

1. construisez ces nouveaux index.
2. Je re-exécute Q_1, combien d'accès sont nécessaires pour avoir une réponse complète ? Numérotez vos appels sur vos index.

Je pose un index composite sur 'has', 'color' et 'legs'

1. construisez ces nouveaux index.
2. Je re-exécute Q_1, combien d'accès sont nécessaires pour avoir une réponse complète ? Numérotez vos appels sur vos index.

4 TinySurvey

L'objectif est de faire un site de sondage en ligne.
Un utilisateur :

— soumet des questions avec la possibilité de répondre par oui ou par non. Une question appartient à une ou plusieurs catégories représentées par des hashtags.
— un utilisateur cherche les derniers sondages selon 1 hashtag,
— un utilisateur répond à un sondage par oui, non mais pas les deux. Il ne peut pas répondre 2 fois
— Il peut voir les messages où il a répondu oui, ou non
— Le site permet de voir les sondages en cours les plus populaires.

Questions : Questions :

1. Décrivez vos entités : les clefs de vos entités, les propriétés de vos entités et les index composites le cas échéant.
2. écrivez une requête GQL permettant d'afficher les derniers sondages contenant un hashtag donné.
3. écrivez une requête permettant d'afficher les derniers sondages où j'ai répondu non.
4. écrivez une requête permettant d'afficher les sondages les plus populaires ie. ceux qui ont le plus de réponses(oui et non confondus)
5. écrivez (en pseudo code) le service REST permettant de répondre à une question
6. est-ce que votre application passe à l'échelle sur la taille des données ? est-ce que votre application passe à l'échelle sur le nombre d'utilisateur ? Est-elle sûre du point de vue de la concurrence ? pourquoi ?