Examen

Pregunta 1 (60 pts)

1. ¿Qué motor de base de datos utilizaría para implementar la navegación entre distintos elementos de información? ¿Es necesario que este motor de base de datos contenga todo el elemento de información o solo palabras clave que permitan establecer relaciones? Justifique su respuesta mediante la elaboración de un pequeño modelo de datos y las relaciones que establecería entre los diferentes elementos de información, lo más importante es garantizar una navegación y que permita descubrir relaciones. (20 pts)

   Yo utilizaría neo4j. Este motor de base de datos se basa en grafos. Estos motores almacenan la información de manera estructurada y establecen relaciones entre diferentes elementos. Con esta estructura, se facilitaría la navegación y descubrimiento de relaciones en la enciclopedia. En Wikipedia, cada artículo suele tener hipervínculos a otros artículos, esto se puede hacer mediante el uso de las relaciones antes mencionadas. Además, estas relaciones se podrían hacer por temas, por autores o por otros factores que tengan en común diferentes artículos, libros o páginas web. Se puede hacer que los nodos sean los libros, artículos o sitios web, mientras que sus conexiones serían todo lo que tengan en común (título similar, autor en común, categoría, fecha de publicación o contenido relacionado).

   (Sinceramente no recuerdo cómo diagramar esto, entonces espero aunque sea ganar unos puntos con la explicación)

2. ¿Qué motor de base de datos utilizaría para almacenar los elementos de información y garantizar full text search? Justifique su respuesta comentando: (20 pts)
   a. Capacidad del motor para implementar full text search.
   b. Particionamiento o sharding de datos.
   c. Representación de elementos de información en la base de datos (tablas, documentos, collections, etc.)

   Primero que todo, se necesita una base de datos que permita full-text search y particionamiento o sharding de datos. Además, yo utilizaría una base de datos con versatilidad a la hora de guardar los datos, es decir. Por último, una base de datos que permita un alto nivel de escalabilidad. Habiendo dicho esto, la base de datos que yo escogería sería ElasticSearch. Elastic posee la capacidad y está diseñada para hacer búsquedas de texto, inclusive si es en múltiples idiomas (lo cual claramente ayudaría a la enciclopedia galáctica puesto que es seguro que esta información se va a conseguir en muchos idiomas diferentes). "Each index in Elasticsearch is divided into one or more shards, each of which may be replicated across multiple nodes to protect against hardware failures." (Size your shards | Elasticsearch Guide [8.8] | Elastic, s. f.), en otras palabras, ya Elastic viene con la capacidad de sharding de base. Por último, debido a que Elastic funciona con documentos JSON, este se vuelve bastante útil a la hora de guardar diversos tipos de datos (puede almacenar libros, artículos científicos y sitios web como documentos separados o colecciones de documentos, dependiendo de las necesidades específicas de la Enciclopedia Galáctica).

3. Describa la forma en la cual combinaría los dos motores anteriores (navegación y full text search) para crear un sistema simple de búsqueda y navegación de información similar al que tiene el sitio

Wikipedia donde se busca un elemento de información y nos podemos mover entre términos. (5 pts)

Primero, se guarda cada libro, artículo científico o página web en un nodo, cada relación entre estos (por ejemplo, un hipervínculo hacia otro artículo) se almacena como un borde entre las conexiones. Luego de añadir el nodo, se indexa esta información en Elastic para luego hacer full-text search (esto contendría los datos necesarios, como el título del artículo y el autor) y el identificador que apunta al nodo. De esta manera, cuando se utiliza el full-text search, el resultado va a apuntar al nodo correspondiente y así se podría hacer la navegación de manera sencilla.

4. ¿De qué forma garantizaría alta disponibilidad de las bases de datos? (5 pts)

   Como se dijo anteriormente, Elastic puede utilizar shards, estos shards a su vez se podrían replicar, para tomar su lugar en caso de cualquier fallo. A su vez, Neo4j permite el uso de replicación, por lo que se puede aplicar lo mismo de ese lado. También se podría utilizar un load balancer para distribuir las consultas entre los diferentes nodos y así evitar, en la medida de lo posible, el colapso de alguno.

5. ¿Cómo podría garantizar que las búsquedas siempre tengan un tiempo de respuesta constante? (5 pts)

   Aunque garantizar que las búsquedas siempre tengan un tiempo de respuesta constante es muy difícil, se podría intentar con el uso eficiente de índices, las técnicas de balanceo y sharding antes mencionadas, con un cache de lo más buscado y mediante el uso de la funcionalidad que muchos cloud services ofrecen, que es un escalado automático de los recursos (esto podría ayudar considerablemente en horas pico).

6. ¿Cómo el uso de caches y localidad podría mejorar el rendimiento del sistema? (5 pts)

   Como ya se mencionó anteriormente, el uso de caches puede mejorar el rendimiento del sistema considerablemente. Más especificamente, se podrían guardar las páginas más visitadas con todos los links asociados para no estar haciendo la misma consulta a la base de datos tantas veces, lo mismo para resultados de las búsquedas más comunes. En cuanto a la localidad, al igual que en los juegos en línea, tener el servidor cerca del punto de acceso es de suma importancia a la hora de obtener las mayores velocidades. Esto se podría combinar con la información necesaria para el cache y hacerlo de manera que se guarde en el cache de cada lugar, los resultados de las consultas más comunes por región.

Pregunta 2 (10 pts)

```
SELECT name FROM artist WHERE name like '%{text}%'

SELECT name FROM album WHERE name like '%{text}%'

SELECT
    a.name as artist_name,
    al.name as album_name,
    s.name,
    s.genre,
    sa.track_number,
    s.length,
    s.description
```

```
FROM artist a
    INNER JOIN artist_song as ON a.artist_id = as.artist_id
    INNER JOIN song s ON as.song_id = s.song_id
    INNER JOIN song_album sa ON s_song_id = sa.song_id
    INNER JOIN album al ON sa.album_id = al_album_id
WHERE
    a.name = '%{name}%'
    AND al.name = '%{name}%'
    AND s.name ='%{name}%'
```

Como administrador o administradora de bases de datos, elabore respuestas a las siguientes preguntas:

- ¿Qué índices definiría para aumentar la velocidad de todo el sistema? Tome en cuenta todos los tipos de índices estudiados en el curso. (3 pts)

  Tomando en cuenta las queries más usuales y problemáticas, haría un índice full-text en la columna "name" de las tabla "artist" y "album" ya que estos están diseñados específicamente para búsquedas de texto usando keywords, especialmente si se hace utilizando "like". Esto optimizaría las primeras dos consultas. En cuanto a la consulta más compleja, esta podría verse beneficiada por la creación de índices B-tree en "name" de las tablas "artist", "album" y "song", esto principalmente por usar el comparador "=" en vez del comparador "like", ya que los índices B-tree están diseñados para comparaciones de igualdad.

- ¿Qué base de datos SQL o NoSQL recomendaría para reemplazar la base de datos actual? Justifique su respuesta. (3 pts).

  Lo primero que hay que tomar en cuenta es que "El patrón de uso es muchas lecturas contra pocas escrituras". Esto nos da un indicio de que podríamos utilizar una base de datos NoSQL. En la documentación oficial de mongodb escribe lo siguiente: "MongoDB indexes use a B-tree data structure.", es decir, los índices de MongoDB utilizan una estructura B-tree, lo cual nos permitiría hacer los antes mencionados índices B-tree. Además de esto, Mongo puede utilizar Full-Text Search, como ya lo utilizamos en el segundo proyecto programado de este curso. Como MongoDB es una base de datos NoSQL que puede utilizar los índices que se dijeron en el punto anterior, esta es la base de datos que yo utilizaría.

- ¿Existirá alguna otra forma de mejorar el rendimiento de la base de datos relacional en especial para la tercera consulta? Comente. (4 pts)

  Además del uso de índices y el cambio de base de datos, se podrían utilizar dos técnicas: particionamiento de datos y utilización de un cache. El particionamiento se tiende a utilizar cuando se tiene una gran cantidad de datos. Al partir las tablas en fragmentos más pequeños, esto distribuiría la carga de las tareas en múltiples nodos, mejorando así el rendimiento de las consultas. El cache se puede utilizar de manera muy efectiva si se saben cuáles son las preguntas más frecuentes; como lo que se consulta no cambia con frecuencia, se puede almacenar las respuestas a las consultas más buscadas en cache, para luego solo devolverlo directamente, sin necesidad de tomar tiempo de procesamiento de más (Ej: se podrían guardar las top 1000 canciones de la plataforma en cache, así, como hay tantas consultas, se disminuiría considerablemente la carga).

Pregunta 3 (20 pts)

Tomando como referencia el diagrama anterior, ¿Cuáles son las buenas prácticas en términos de seguridad que se deben seguir cuando se instala un motor de base de datos en el Cloud? Fundamente su respuesta hablando de la seguridad de cada uno de los componentes que se exponen en el diagrama.

En este diagrama se puede ver una de las primeras recomendaciones: la segregación de datos. Como se puede ver, se separan los componentes en diferentes redes. Esto nos permite monitorear y restringir el acceso a los diferentes componentes. Si todo estuviera en una misma red, bastaría con penetrar la seguridad de una red y se tendría acceso ilimitado a todo. Además, la encriptación de datos se puede aplicar en todas las capas mostradas en el diagrama: en la red pública con protocolos como HTTPS, en la red privada con encriptación a nivel de base de datos y en la red de almacenamiento con encriptación a nivel de disco. Por último, el acceso utilizando usuarios federados también se puede aplicar a todas las capas.

1. Capa aplicación en la red pública

    - contraseñas seguras: utilizar combinaciones complejas de letras minúsculas, mayúsculas, símbolos y números, además de esto deberían de ser únicas y cambiarse cada cierto tiempo.
    - 2 factor authentication (2fa): tal y como se utiliza en muchas páginas y aplicaciones hoy en día, tener una manera de verificación a través de un dispositivo personal (la mayoría del tiempo un celular o similar) o un correo.
    - control de acceso basado en roles: se debe usar el Principle of Least Privilege (POLP), es decir los usuarios solo pueden hacer lo que se supone que deben hacer, se les da la mínima cantidad de privilegios para que realicen su trabajo.
    - Se puede implementar un sistema de registro de eventos que pueda detectar comportamientos sospechosos en la aplicación.

2. Instancia/motor de base de datos en la red privada

    - Actualizaciones y parches: se tiene que asegurar que se tenga todas las últimas updates de seguridad y verificar esto constantemente.
    - Firewall: restringir el acceso a solo IPs específicos o rangos de IPs confiables.
    - Se puede configurar una vpn para evitar el acceso de usuarios no permitidos a través de redes públicas.
    - Tal y como en la capa 1, se puede implementar un registro de eventos para detectar comportamientos sospechosos.
    - Copias de seguridad y recuperación: se puede implementar un plan de backups automatizado que sean almacenados de forma segura y estar disponibles en caso de pérdida de datos y/o corrupción de estos.

3. Disco en la red de almacenamiento: en esta capa se habla de los data centers, por lo que es necesario que estos sean seguros y que estén disponibles todo el tiempo.

    - Al hablarse de almacenamiento físico de los datos, se requiere el uso de buen material para evitar problemas físicos de los discos.
    - Guardias de seguridad y vigilancia constante.
    - Hay diversas tecnologías más avanzadas que se usan en diversos data centers, como por ejemplo reconocimiento facial y accesorios como brazaletes especiales.
    - Las zonas geográficas poseen replicación con bypass de internet.

    - Usar UPS de gran capacidad y utilizar equipos con garantía, como los discos iSCSI.

Pregunta 4 (10 pts)

La Observabilidad es una gran herramienta que nos permite tener una visión en el tiempo de la forma en la cual se comportan sistemas computacionales, estos sistemas hacen uso extensivo de bases de datos de series de tiempo, una de las más utilizadas es Prometheus, pero existen soluciones que utilizan otras bases de datos o motores de búsqueda como Elasticsearch u OpenSearch. Como ingeniera o ingeniero a cargo de los sistemas de Observabilidad de una empresa, se le ha solicitado dar respuesta a las siguientes preguntas, con el fin de determinar la estrategia que seguirá la empresa en términos de Observabilidad en los siguientes años.

- ¿Por qué las bases de datos de series de tiempo son tan utilizadas en soluciones de Observabilidad? Realice un análisis desde el punto de vista de la naturaleza de los datos que se recolectan. (2 pts)

  La observabilidad consiste en estar analizando el funcionamiento del sistema en intervalos regulares o tiempo real. Las bases de datos del tipo time series están diseñadas para manejar datos ordenados según su timestamp. Además de esto, este tipo de base de datos está optimizado de tal manera que puedan mantener una gran cantidad de información y seguir siendo eficientes para el tipo de consultas que se necesita.

- ¿Es posible utilizar BigTable como una base de datos de series de tiempo que se pueda utilizar como parte de una solución de Observabilidad? Justifique su respuesta desde el punto de vista de la naturaleza de la base de datos. (2 pts)

  BigTable es una base de datos diseñada de manera que pueda manejar de manera óptima datos no estructurados. La observabilidad se basa en datos estructurados y organizados basándose en el timestamp. Esto no quiere decir que no se pueda utilizar en este caso, sin embargo, no sería la mejor opción. Saldría mucho mejor en cuanto a costos utilizar una base de datos especializada en este tipo de datos.

- Suponiendo que tenemos una solución de Observabilidad que utiliza Elasticsearch, ¿Cómo podemos ahorrar dinero con información histórica? (2 pts)

  Lo primero que se podría hacer es hacer un análisis detallado de qué tan antigua se necesita realmente la información y eliminar o archivar todo lo que sea más antiguo que eso. En clase se vio una de las técnicas de compresión que tiene elasticsearch (cuando comprimimos los json a parquet), esta es otra manera, no tan drástica de optimizar los datos históricos, ya que elastic trae consigo la capacidad de diversos tipos de compresión. Esta compresión se podría aplicar en vez de la eliminación o archivación de los datos antes mencionados. Otra técnica que se podría utilizar es la del Rollover de índices que es mencionada en la documentación oficial de elasticsearch, lo que consiste en reescribir índices de manera que se divida en múltiples índices más pequeños.

- Comente las ventajas y las desventajas de utilizar un servicio de Observabilidad on-premise (por ejemplo, Prometheus y Grafana) vs un Managed Service (como Datadog), justifique su respuesta con la experiencia obtenida en la tarea corta 1 de este curso. (4 pts)

  Uno de los principales problemas que nuestro grupo tuvo con la primera tarea corta de este curso fue lo que se duró configurando y haciendo que un servicio de observabilidad on-premise funcionara. Esto no solo toma tiempo, sino que toma recursos que podrían ser utilizados en otro lugar. Una de las

ventajas más grandes del cloud computing es el hecho de poder delegarle una tarea como esta a una compañía o sistema que se encargue. Además de esto, un Managed Service puede proveer otros servicios con gran escalabilidad para nuestra aplicación.

Referencias

- Rollover | Elasticsearch Guide [8.8] | Elastic. (s. f.). Elastic. https://www.elastic.co/guide/en/elasticsearch/reference/current/index-rollover.html
- Indexes — MongoDB Manual. (s. f.). https://www.mongodb.com/docs/manual/indexes/
- Size your shards | Elasticsearch Guide [8.8] | Elastic. (s. f.). Elastic. https://www.elastic.co/guide/en/elasticsearch/reference/current/size-your-shards.html