Examen

Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Bases de Datos II (IC 4302)
Segundo Semestre 2022

Gabriela Gutiérrez Valverde - 2019024089

Pregunta 1:

1. Dar una **solución detallada de cómo podría mejorar el rendimiento** de la base de datos actual, reduciendo el downtime al mínimo, esto permitirá ganar tiempo para dar una solución mucho más duradera con la mínima afectación a los usuarios.

Para reducir el downtime y mejorar el rendimiento de forma temporal, se podría normalizar, pero en este caso creo que no es necesario (al menos por el momento) porque podría terminar afectando el rendimiento. Lo mejor sería tener una réplica de la base de datos, pero almacenada en un lugar distinto a la primera, de esta forma podemos mitigar el downtime, y que la réplica nos ayude con las consultas, así los usuarios podrán tener una mejor velocidad de respuestas. Y podríamos adaptar un modelo Master-Master con sincronización de datos.

2. Dar una **recomendación detallada de que tipo de base de datos se debería utilizar** para abordar este problema, además debe recomendar algunas de las bases de datos SQL o NoSQL estudiadas durante el curso tanto en lecturas, así como las utilizadas en proyectos o ejemplos en clase. Tome en cuenta que sería posible utilizar más de una base de datos para optimizar el almacenamiento de los datos de la tabla post, amigos y usuario, tome en cuenta que tan fácil es escalar la base de datos en su recomendación, debe dar prioridad a servicios managed services y SaaS, no olvide la localidad y naturaleza de los datos.

Primero antes de almacenar los datos haría una normalización sobre la tabla usuario, ya que como está en el momento si tenemos un millón de usuarios, tendríamos un millón de países y muchos más estados, si lo sacamos podemos reducir a solamente máximo 194 registros y los estados en los que se encuentren usuarios de la red social. También se podría hacer un análisis sobre qué tan conveniente podría ser sacar también los apellidos, ya que al menos en el caso de Costa Rica suelen repetirse

mucho. Ya una vez con las mejoras hechas sugeriría que se siga utilizando un sistema SQL para el manejo de los usuarios, ya que altamente consistente, no va a permitir que se creen dos usuarios con los mismos datos. Una base de las lecturas que recomendaría para esto es Spanner, ya que alta consistencia, réplicas globales y alta disponibilidad, además ofrece algunas otras características que las bases de datos SQL convencionales no ofrecen.
Para los post, lo mejor es utilizar una base de datos de documentos, que utilice JSON, de esa manera no hay problema si se ingresa o no un hashtag, entonces una base de datos NoSQL sería lo ideal, ofrecen alto rendimiento para lecturas, se pueden alojar fácilmente en la nube y son escalables, específicamente utilizaría ElasticSearch ya que esta base cumple con todos los requisitos mencionados y además ofrece herramientas si quisiéramos obtener insights de los datos. En este caso no es necesaria la alta consistencia ya que, no hay problema si 2 o más personas responden a un post al mismo tiempo y segundo un usuario tendría una sola sesión abierta por lo que no va a editar el mismo post dos veces.
Para solucionar un problema como el de los mensajes de odio o agrupar las historias para mostrarle al usuario se podría utilizar un sistema como el de neo4j, que no solamente tiene los datos, sino que crea conexiones sobre estos. Para los mensajes de odio se podría crear una rastreabilidad para lograr tener toda la frase y poder revisar o eliminar todos los postings que las tengan. Además de que ofrece el “real-time recommendations” qué es útil ya que le puede mostrar al usuario el conjunto de post que forman una frase que le puede gustar.

3. **Comente acerca de que tan conveniente es mantener la base de datos actual en la casa de uno de los fundadores**, comparado con mover ésta algún Cloud Provider como AWS.

Es 0 conveniente, puede causar que pase mucho tiempo el servicio caído por condiciones que afectan solamente la casa donde se encuentra la base de datos, como un apagón, conexión de red, incluso en el peor de los casos podría haber un incidente mayor como una inundación o un incendio y se perdieron todos los datos, y esto puede significar el fin para una aplicación como esta. Al tener el servicio con un Cloud Provider estos aseguran alta disponibilidad, varía según el servicio pero normalmente son no más de unos minutos de downtime, además (dependiendo del contrato) las empresas que dan estos servicios suelen hacerse responsables si se excede el tiempo o si existe alguna pérdida de datos.

4. Basándose en el funcionamiento de un índice invertido el cual fue estudiado en clase y es utilizado por motores como Elasticsearch y el concepto de Natural

Languaje Processing (NLP) llamado Stemming el cual también fue discutido en clase, comente **¿Cómo se podría reducir el memory footprint de la base de datos actual?**

Podría reducir el memory footprint por la característica que se llama Streaming que es conservar la raíz de la palabra. Primero, se le asigna una relevancia, lo cual para una red social es útil porque siempre van a existir tendencias, esto nos permite tener más a mano los índices más consultados y para mejorar el memory footprint se puede tener los token con menos relevancia almacenados tipo warm o cold. Luego, ayuda también al eliminar los caracteres sin relevancia.

Pregunta 2

1. Comente, **¿Cómo afectan los índices en el rendimiento de las bases de datos relacionales?**, enfoque su respuesta tanto en como benefician el rendimiento así la forma en la cual lo impactan de forma negativa.

Los índices generalmente impactan el rendimiento de forma positiva ya que mejoran la duración de las búsquedas, esto porque crea un ordenamiento de los datos que puede ser un árbol, un hash o con diccionario y posting list. Pero hay restricciones, primero, las columnas que sean incluidas en el índice deben ser poco modificadas, esto porque cuando se modifican los datos se debe reordenar el/los índice(s). Luego, al tener muchos índices cae el rendimiento, porque se aumenta la cantidad de memoria que se tiene reservada, aunque no se esté utilizando (memory footprint). Por último se debe tener cuidado con la programación y las consultas que se hacen a los índices, si se hace un mal uso de include se va a generar más memory footprint y si las consultas incluyen expresiones como “ORDER BY” o “JOIN” se tiene un impacto muy negativo en el rendimiento ya que se debe ordenar las columnas para hacer la búsqueda.

2. Suponiendo que el hardware no es un problema (se puede comprar cuanto se necesite), **¿Podemos crear cuantos índices queramos o estos no tendrán mayor impacto en el rendimiento?**

Aún teniendo el mejor hardware no sería lo recomendable crear “cuantos índices queramos”, porque seguiría habiendo impacto negativo en el rendimiento, cuando se podría utilizar más bien para agilizar los tiempos de respuesta, lo mejor es realizar un análisis de las consultas más frecuentes, las tablas que más se actualizan e intentar hacer los índices que sí nos puedan funcionar, en caso de que no se cumpla con las características para el índice lo mejor es buscar otro tipo de ordenamiento o incluso ver si la base que utilizamos es realmente la más óptima para el trabajo.

Pregunta 3

El rendimiento de todo sistema de base de datos puede verse afectado por muchos factores, uno de ellos es el ambiente en el cual se ejecuta, este se encuentra compuesto por los componentes de hardware y el sistema operativo y otros programas de usuario compitiendo por los recursos del computador. Comente de forma clara y concisa, **¿Cómo afecta el rendimiento de una base de datos los componentes ilustrados en la Figura 1?**

Comenzando por el **disco**, este es el componente que se encarga del almacenamiento y esto afecta ya que también es el componente más lento, lo que puede causar mal rendimiento. Ligado a esto, tenemos el **Sistema Operativo**, que es el que interactúa con el disco, es decir, si la aplicación pide algún dato de disco que necesita en memoria principal, es el sistema operativo quien decide como lo trae y cuando (porque puede ser que haya otra aplicación que hizo la solicitud primero), y estos datos se traen por medio de un “bus” y este tiene un límite que puede traer cada vez. Los **programas de usuario** pueden interferir con la base de datos ya que ocupan espacio en memoria, pueden también hacer solicitudes al SO y traer datos de disco esto baja el rendimiento. La **memoria principal** es mucho más rápida que el disco pero sus datos no son persistentes, por lo que si ocurre un evento (como por ejemplo, un apagón) los datos se perderán, además tiene menos espacio que el disco por lo que solo cierta cantidad se puede almacenar y se debe estar devolviendo y trayendo datos. Luego está el **CPU** que es el componente más rápido de la computadora, la capacidad de la Base de Datos depende de este componente por la cantidad de núcleos, hilos y la velocidad a la que se pueden ejecutar los juegos de registros. Y finalmente está el **caché** que comunica la memoria principal y el CPU, tiene la función de una memoria de alta velocidad pero contiene una cantidad muy limitada de datos, es muy útil si generalmente se llama a los mismos datos, pero si se llaman datos aleatorios puede hacer que más bien haga más lentas las respuestas.

Pregunta 4

La escalabilidad automática es una característica muy deseada en los sistemas de bases de datos tanto SQL como NoSQL, la misma permite mediante la obtención de métricas en tiempo real interpretar el comportamiento actual para predecir el comportamiento futuro, con esto se puede ajustar tanto el hardware como la configuración de las bases de datos, para poder atender el workload de un sistema. **Comente la importancia de la Observabilidad** tanto a nivel de aplicación como de

base de datos para lograr una escalabilidad automática adecuada, **¿Considera que las métricas de memoria, CPU y disco son suficientes para lograr ésta?**

- Para esta respuesta me voy a apoyar con la definición de Observabilidad del artículo de "Observabilidad: Obtenga información y mejore el rendimiento de sus aplicaciones, usuarios e infraestructura" de AWS

Como se menciona en el artículo, "la observabilidad describe lo bien que se puede entender lo que ocurre en un sistema, a menudo mediante instrumentos para recopilar métricas, registros o rastreos". Esto quiere decir que cuando aplicamos observabilidad a una aplicación (entiéndase conjunto de programas, bases de datos, pods, etc) podemos conocer el rendimiento que tiene en diferentes áreas o componentes, como sería por ejemplo, ver la interacción con la memoria. Además con el monitoreo podemos sacar conclusiones sobre las métricas obtenidas. Y el objetivo es "recopilar y analizar los datos de las aplicaciones y la infraestructura para que pueda comprender sus estados internos y recibir alertas a fin de solucionar y resolver problemas con la disponibilidad y el rendimiento de la aplicación para mejorar la experiencia del usuario final." según AWS.
A nivel de aplicación, podemos determinar si existen horas o días en específico que exista más tráfico, puede ser que algún componente genere un cuello de botella y sea necesario generar una solución como podría ser por ejemplo, generar más réplicas en caso de que se trabaje con pods. A nivel de base de datos, nos puede ayudar a optimizar el manejo de los datos, ver si estuvo abajo por algún momento e incluso poder determinar qué causó esto. Y aprender sobre las métricas que recibimos es lo que permite automatizar la escalabilidad, sabiendo cuando es más necesario contar con más recursos para no afectar el funcionamiento normal.
En cuanto si las métricas de memoria, CPU y disco son suficientes: podrían ser suficientes en una aplicación local con muy pocos usuarios, pero ya un nivel superior definitivamente no. Es importante medir los componentes de la aplicación, es decir, generar también métricas de las bases de datos, de los pods, las colas, y demás componentes que se utilicen. Medir la cantidad de usuarios que ingresa, las horas y días en que hay más usuarios, el tráfico que se genera en la red, entre otros componentes que pueden ser útiles dependiendo de la aplicación.