Examen

Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Bases de Datos II (IC 4302)

Segundo Semestre 2022

Estudiante: Andrea María Li Hernández

Carné: 2021028783

Pregunta 1 (60 pts)

Time No More es la red social más pequeña jamás creada, es conocida como **one word social network (OWSN)**, este término fue acuñado por un profesor mientras diseñaba un examen, el funcionamiento es muy simple, como en cualquier otra red social las personas pueden tener amigos y postear mensajes, pero en este caso son simples mensaje con una palabra. En sus primeros 2 meses, Time No More logró reunir cerca de 500 mil usuarios, con un volumen de queries de escrituras (nuevos posts) de 5 posts por segundo y un volumen de queries de lectura de 100 queries por segundo.

Originalmente esta OSWSN fue creada en una base de datos relacional, se puede asumir que es un motor MariaDB standalone en se encuentra la casa de uno de los fundadores, esto está causando muchos problemas de rendimiento y se estima que en menos de 4 meses si el crecimiento se mantiene como en los dos primeros meses la base de datos colapsará.

Esta red social, aunque simple se ha hecho popular, ya que permite entre otras cosas, mostrar el estado de ánimo, exponer deseos, aspiraciones, y otros, todo en una sola palabra, lo cual es sumamente conveniente para hacer análisis de mercadeo y además simplifica los modelos de inteligencia artificial.

También entre la juventud se hizo muy popular enviar mensajes palabra por palabra o hasta construir historias, donde cada persona escribe una palabra con un hashtag (máximo 1) y mediante el tiempo de inserción en la base de datos, se va construyendo

una historia, esto ha sido utilizado por grupos con malas intenciones para enviar mensajes inapropiados o coordinar grupos de odio, que normalmente evitan enviar el hashtag para evitar ser descubiertos, al ser una base de datos relacional y al ser una característica muy utilizada, la base de datos relacional no está funcionando correctamente ya que los queries duran mucho tiempo.

Cada registro de la base de datos Time No More, tiene el siguiente formato:

Tabla **post**:

id	timestamp	hashtag	palabra	id_usuario	latitud	longitud

Tabla **usuario**:

id	nombre	apellidos	telefonos	pais	estado	zip

Tabla amigos:

id_usuario1	id_usuario2

En calidad de CTO de la compañía Timeouts No More, ustedes han sido contactados o contactadas para ayudar a Time No More e intentar solucionar el problema, por esta razón debe realizar las siguientes tareas:

- 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. (10 pts)

El enfoque que le daría a esta solución es distribuir y particionar la base de datos actual. Empezando por la distribución, que el único servidor se ubique en la casa de uno de los fundadores pone en alto riesgo al sistema, porque al estar centralizado en una sola máquina, considerando el peor de los casos, si esta falla se pasaría de tener alto *downtime* a tener un total cierre del servicio. Por esto, para la solución se debería de

realizar un estudio para determinar cuáles son las mejores ubicaciones para poner más servidores y, seguidamente, instalar al menos tres bases de datos más en los diferentes lugares encontrados.

Luego de distribuir la BD, se realiza el particionamiento, con esto se pasaría de tener un solo servidor con un masivo volumen de datos, a diferentes servidores que pueden trabajar de forma paralela y con conjuntos de datos más pequeños; lo que puede reducir el tiempo de respuesta, agilizar la búsqueda de datos y minimizar el daño en caso de una emergencia.

Cabe recalcar que debido a que se está trabajando con una base de datos relacional, de momento no recomendaría implementar *Federated Queries*, pues considero que podrían penalizar el rendimiento de la BD porque si una BD relacional ya consume tiempo asegurando la consistencia de los datos y la integridad referencial, sumarle el tiempo que puede tardar el sistema en unir los resultados de los diferentes servidores únicamente aumentaría el tiempo de espera para el usuario.

- 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. (30 pts)

Primero, recomendaría utilizar la base de datos NoSQL ElasticSearch para almacenar la tabla **post** por las siguientes razones:

  - Se puede utilizar como una base de datos de series de tiempo, lo que favorece a las publicaciones de la red social que poseen el atributo

timestamp, además, esto nos brinda mayor precisión para la captura de la fecha y hora a la que se publicó el mensaje.

- Dado que ElasticSearch posee consistencia eventual, se obtiene una mayor disponibilidad de los datos para satisfacer la carga de trabajo de lectura que posee el sistema y se pueden retornar rápidamente los resultados de dichas lecturas.
- Se pueden definir data tiers que favorezcan a la localidad de los datos, recomendaría utilizar Cold para almacenar las publicaciones más antiguas, el criterio que se utilice para clasificar las publicaciones debe ser definida por Time No More; una opción podría ser que las publicaciones con una antigüedad de 6 meses pasen a este data tier. La mayor ventaja es que se utilizaría almacenamiento más barato y se necesitaría menos memoria. Por otro lado, los posts más recientes (tal vez los del mes o de la semana), se pueden almacenar en el data tier Hot, que satisface la alta demanda de consulta con potencia en CPU, disco y memoria.

Luego, para la tabla de **usuarios** y la tabla **amigos** recomendaría utilizar la base de datos NoSQL MongoDB por las siguientes razones:

- La información de los usuarios se puede almacenar en archivos JSON, los cuales permiten el rápido acceso a la información y no consumen mucho almacenamiento; este último aspecto es beneficioso pues la red social presenta un rápido crecimiento en cantidad de usuarios.
- MongoDB ofrece gran flexibilidad a la hora de definir la consistencia que va a tener la base de datos, por lo que se podría definir alta consistencia para la información de ambas tablas.
- MongoDB favorece a los sistemas que necesitan de escalabilidad gracias a los *shards*, que son básicamente *chunks* de datos, y las réplicas que es donde se almacenan. Esto es esencial para que se pueda satisfacer el volumen creciente de usuarios.

Como **managed service** recomendaría Bigtable, este puede ser beneficioso ya que implementa *locality groups,* los cuales permiten agrupar columnas y obtener mejor localidad de los datos, por ejemplo, las columnas resaltadas en la tabla usuarios, que todas se relacionan con la ubicación del usuario:

id	nombre	apellidos	telefonos	pais	estado	zip

Además, Bigtable es fácil de ampliar la capacidad de los *clusters*, porque el sistema simplemente agrega más máquinas.

Como **SaaS** recomendaría fuertemente que se utilice un Graph Data Platform como Neo4j, porque permite optimizar *datasets* que se encuentran interconectados. Además, la naturaleza de una red social son las relaciones entre los usuarios y el contenido que publican. Neo4j permite realizar queries y visualizar fácilmente los datos que están conectados y los patrones que se pueden presentar. Algo importante es que Neo4j también ayudaría a identificar rápidamente, mediante la trazabilidad, a los usuarios partícipes de cadenas mal intencionadas y así tomar acción contra estos y sus publicaciones.

- 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. (10 pts)

Mantener la base de datos en la casa de uno de los fundadores vuelve sumamente vulnerable a la BD, porque al tener un único punto de fallo, si ocurre un error se deshabilitaría por completo el servicio; además de que también se encuentra más expuesto en términos de seguridad porque con solo hackear ese servidor, tendrían el acceso a todo el sistema.

Esto también es inconveniente para el rendimiento de la base de datos, debido a que se tiene un gran volumen de datos en un solo servidor y no se está tomando en cuenta la regionalización de los datos, donde los usuarios deberían de poseer más cercanía a

los servidores y así un servicio más rápido. Por esas razones es recomendable que la base de datos emigre a un Cloud Provider como AWS, donde se pueden configurar las regiones de los servidores, la empresa debe de preocuparse menos por la definición e instalación del hardware a utilizar y se abre la posibilidad de distribuir la base de datos, lo cual tendría un impacto positivo en el rendimiento del sistema.

- 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? (10 pts)

Con un enfoque en la indexación de la tabla post, en la base de datos actual se podría hacer lo siguiente:

  - Cuando un usuario realice una búsqueda de una palabra, se aplicará Stemming para obtener la raíz de la palabra a buscar y, una vez hecho esto, se procede con la búsqueda mediante el índice invertido.
  - El índice invertido se puede aplicar de tal forma que se tiene una relación entre la palabra y la identificación de aquellas publicaciones donde aparece. A estas publicaciones se les podrían asignar un puesto con un algoritmo que se base en qué tan reciente es la publicación. Entre más reciente, mayor relevancia.
  - Finalmente, el resultado serían las publicaciones más relevantes que coincidan con la búsqueda.

Esto permitiría reducir el memory footprint de la base de datos porque al tener un índice más pequeño, obtenemos velocidad y el índice ocuparía menos espacio en memoria. Además, al no tener que realizar búsquedas exactas y exhaustivas, pues estaremos buscando por relevancia y no exactitud; podemos disminuir el consumo de memoria mientras el sistema está funcionando.

Pregunta 2 (10 pts)

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.
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?

Si se analiza el caso de uso, la frecuencia con que se modificarán los datos y el patrón de acceso a estos, entre otros factores; se puede definir correctamente el tipo de índice que se necesita, lo que brinda varios beneficios para la base de datos relacional. Primero se puede reducir el memory footprint de la aplicación, lo que representa un buen manejo de los recursos y ahorro de dinero. También se puede lograr el acceso rápido a la información, lo que significa menos tiempo de espera para el usuario y mejor rendimiento de la base de datos.

Los índices pueden impactar de forma negativa si se implementa uno que no se ajusta al caso de uso, se crean en gran cantidad o cuando se vuelven muy grandes. Implementar uno que penaliza al caso de uso ralentiza las búsquedas y consume recursos de forma innecesaria, lo que significa que el usuario debe esperar más a que se procesen sus consultas, lo que refleja un bajo rendimiento de la base de datos.

Suponiendo que se tiene el presupuesto para comprar todo el hardware que se desee, aún así no es recomendable crear una cantidad desmesurada de índices, dado que más cantidad significa mayor memory footprint y penalización en el rendimiento de la base de datos. Con muchos índices se podría llegar a tener redundancia en las búsquedas, lo que sumará al tiempo de espera del usuario.

Pregunta 3 (20 pts)

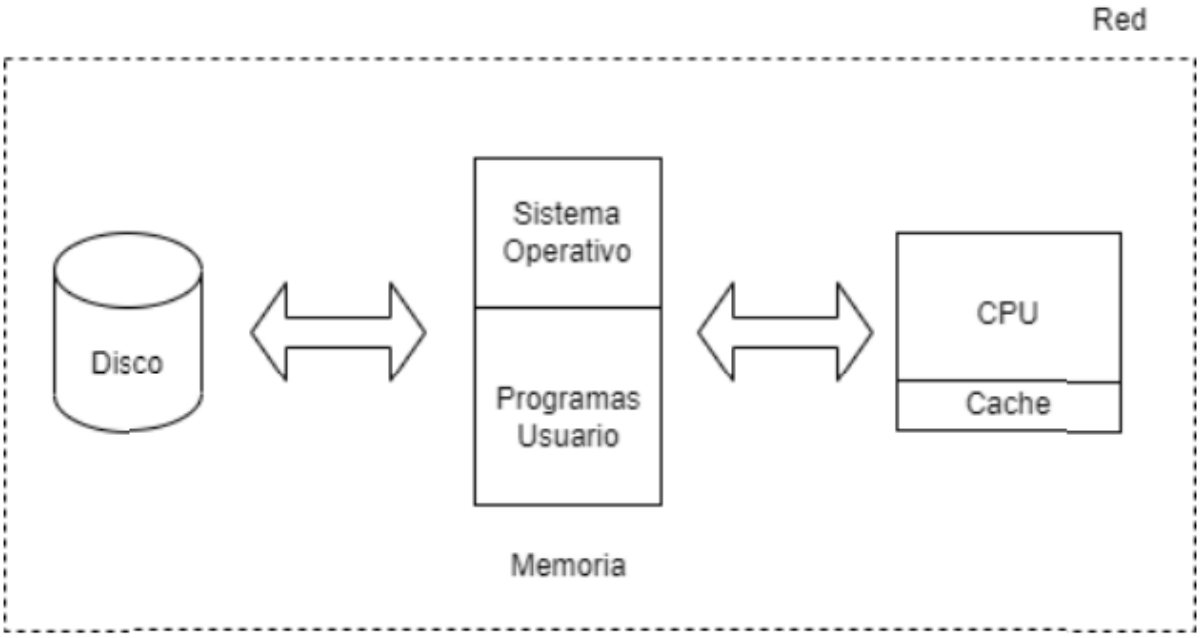

Figura 1

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?

Cuando no se encuentra la información en caché o si este es muy pequeño, se debe consultar en la memoria principal, lo que genera *overhead* y penaliza el rendimiento de la base de datos. Si no se tiene la suficiente memoria para satisfacer la carga de trabajo del sistema, habrá un aumento de los *timeouts* para cada usuario.

El disco es de los componentes que definirán las operaciones de entrada y salida por segundo y si se debe paginar mucho disco también se aumenta el memory footprint

y la cifra mencionada anteriormente se ve afectada. Si el CPU posee pocos núcleos, perdemos la posibilidad de paralelizar el trabajo en la base de datos y aumentar la eficiencia de la aplicación.

La red nos define aspectos como la sincronización de las bases de datos y la cantidad de clientes que el sistema es capaz de atender, si no se pueden sincronizar correctamente perdemos consistencia de los datos y si no se puede atender a todos los usuarios estos tendrían que enfrentar largos periodos de *timeout* o, en el peor de los casos, que el sistema colapse y se tenga *downtime*.

Por último, el sistema operativo es el encargado de brindarle una ventana de tiempo a cada programa de usuario, y si se tiene un SO monolítico se encuentran desventajas como poca flexibilidad, entonces no se puede administrar de forma óptima los recursos lo cual sería muy útil cuando hay varios programas compitiendo por los recursos; y si algo falla en el sistema, puede tener un impacto en el rendimiento en su totalidad. También se tienen los cambios de contexto, donde los procesos son interrumpidos por el SO.

Pregunta 4 (10 pts)

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?

Son importantes pues es gracias a estas métricas que se puede automatizar la escalabilidad, si no se tuvieran dichos datos a disponibilidad; manualmente y cada cierto periodo de tiempo las personas encargadas deben de analizar el comportamiento y tomar decisiones para determinar de qué manera se va a ajustar la base de datos y el equipo

para beneficiar al negocio. Este proceso consume más tiempo y esfuerzo, además, se tendrían resultados menos acertados gracias al factor humano y debido a que no se está contemplando el comportamiento del sistema durante todo su horario de funcionamiento.

Por otro lado, métricas de memoria, CPU y disco **no son suficientes** para lograr una escalabilidad automática, existen otros factores que influyen en la toma de decisiones para el ajuste del sistema. Por ejemplo, a nivel de aplicación, se pueden obtener métricas como la cantidad de usuarios que ingresan al sistema, los horarios con más y menos tráfico, tiempo promedio que tarda el sistema en procesar una consulta y devolver el resultado al usuario; entre otros.

Mientras que con las métricas de memoria, CPU y disco se puede estudiar el **comportamiento del *hardware***, con las métricas mencionadas anteriormente se puede analizar adecuadamente el **comportamiento de los usuarios**. Por ejemplo, analizando los horarios con diferentes niveles de tráfico, se pueden agregar más nodos o reducirlos en el caso contrario, y así ahorrar dinero y recursos. Otro caso puede ser que se observe que el sistema está tardando más de lo esperado para procesar una consulta, lo que significa que hay que comprar mejor *hardware* que sea capaz de disminuir ese tiempo de espera para el usuario.