Lectura 4 y 5: Prometheus & Grafana

Diego Granados Retana 2022158363

Bases de datos II

19 de setiembre, 2023

Responda las siguientes preguntas:

¿Cómo se diferencia el modelo de datos de Big Tables de una base de datos SQL?

Bigtable se asimila mucho a una base de datos, pero tiene varias diferencias. No tiene un modelo de datos relacional completo, tiene clientes con datos simples que permiten el control dinámico del diseño y formato. Las filas y columnas tienen nombres que son strings arbitrarios. Los datos se tratan como strings no interpretados, aunque es posible pasar estas a formas estructuradas y semiestructuradas. Los clientes pueden controlar la localidad de los datos y controlar dinámicamente si pedir la información de la memoria principal o del disco. Un Bigtable es un mapa distribuido, persistente, multidimensional y ordenado. Cada mapa se indexa con una clave de fila, una clave de columna y la fecha y hora. Cada valor en el mapa es un arreglo de bytes sin interpretar. Esto se diferencia a una base de datos SQL porque en la SQL, los valores almacenados deben tener definido su tipo de datos y tamaño. Como las bigtables guardan solo strings, pueden almacenar cualquier información que es después utilizada por el usuario para sus objetivos. Además, esto implica que la base de datos no tiene que hacer mucho procesamiento de la información, solo almacenarla. Otra diferencia es que las claves de las filas y columnas pueden ser cualquier cosa como un URL o nombres de características.

¿Cuáles decisiones de diseño en Big Table aumenta el rendimiento del sistema? Explique.

Primero, como los valores se almacenan como strings sin interpretar, esto elimina la validación del tipo de datos que se está insertando a una columna específica. Seguramente nada más hay que validar si es un string o no. Segundo, los bigtables mantienen las llaves de filas en orden lexicográfico, por lo que las búsquedas son más fáciles. Tercero, los rangos de filas de una tabla se particionan dinámicamente. Cada rango es una tableta, la unidad de distribución y balanceo de carga. Esto causa que lecturas de rangos cortos sean muy eficientes y requieran pocas máquinas. Si se escogen bien las filas de las tabletas, se puede obtener buena localidad de los datos y prevenir más comunicación entre máquinas. Las llaves de columnas se agrupan en conjuntos llamados familias. Los datos en una familia son del mismo tipo. Las columnas se tienen que agrupar antes de que se pueda almacenar datos en alguna de esas columnas. Las familias usualmente no cambian durante la operación. El control de acceso, disco y memoria se hace a nivel de las familias de columnas. Esto permite que se pueda añadir nuevos datos base, leer datos base y crear familias de columnas derivadas. Se utilizan marcas de tiempo para ordenar las celdas para que las más recientes se puedan leer de primero. Hay configuraciones por columna que permiten a Bigtable eliminar versiones de celdas que ya no se necesitan, como que solo se necesita un número de versiones o que la versiones no pueden tener más de cierto tiempo. Uno puede agrupar familias de columnas en un grupo de localidad. Esto permite lecturas más eficientes. Se pueden comprimir los SSTables de cada grupo de localidad. Se pueden tener datos en la cache. Hay dos niveles. La cache de escaneos tiene los pares de llaves y valores que retorna un SSTable al código del servidor de tabletas. La cache de bloques almacena los bloques de las SSTables que fueron leídos del Google File System. Se pueden poner filtros de Bloom, que nos permiten saber si una tabla tiene datos para una fila y columna

específicas. Se usa un solo log de commits por servidor de tabletas. Se ponen operaciones de diferentes tabletas en el mismo log. Esto da mucho mejor rendimiento, pero dificulta la recuperación de los datos, ya que se tendría que leer el log la cantidad de tabletas que haya. Esto se arregla al ordenar el log con base en las tablas y filas. Escribir en el log puede causar problemas, entonces cada servidor tiene dos hilos de escritura, nada más que solo uno se usa a la vez. Finalmente, las SSTables son inmutables, por lo que la concurrencia es muy eficiente.

¿Considera que Big Table podría cumplir el papel de Prometheus en un sistema de Observabilidad? En caso de responder No, explique detalladamente, en caso de responder si, ¿utilizarían versiones de timestamps para cada métrica y recolectarían cada métrica como un row separado?

Yo creo que Bigtable sí podría servir para almacenar la información que almacena Prometheus. Me parece que utilizar la versión de timestamps sí ayudaría mucho a simular el efecto de un time series, ya que las versiones se almacenarían de tal modo que las versiones más recientes se lean primero. No obstante, considero que tal vez las métricas se podrían usar como columnas también y que cada fila sea una base de datos o sistema que se está monitoreando. De esta forma, todas las métricas de un sistema se almacenarían en un solo registro y solo se tendría que consultar ese para obtenerlas. Luego, se obtiene cada versión de una métrica y se puede realizar una gráfica.

Explique en detalle la organización de tablets en Big Table

En Bigtable, hay tres componentes principales, una librería que está conectada a todos los clientes, un servidor maestro y muchos servidores de tabletas. El maestro asigna tabletas a los otros servidores. Un servidor tiene de 10 a 1000 tabletas. Bigtable utiliza un servicio llamado Chubby, que un sistema distribuido de locking. Este se utiliza para asegurarse que hay un solo servidor maestro, descubrir servidores de tabletas, almacenar información del esquema de bigtable y para almacenar listas de control de acceso. Las tabletas se almacenan en una jerarquía similar a un árbol B+. En Chubby se almacena una tableta raíz, que contiene la ubicación de todas las tabletas en la tabla de Metadata. Esta nunca se puede partir para asegurarse que la jerarquía nunca tenga más de tres niveles. La librería almacena las ubicaciones de las tabletas en la caché. Si no sabe a dónde está una, busca su ubicación recursivamente por la jerarquía. Una tableta se asigna a solo un servidor a la vez. Los servidores de tabletas necesitan que exista un directorio específico para ellos en Chubby. Si no existe, se eliminan y se libera el lock en el directorio que tenían. De esta forma, el servidor maestro puede volver a reasignar las tabletas. Las tabletas que no están en un servidor se agrupan en un conjunto de tabletas no asignadas. El conjunto de tabletas existentes solo se puede modificar cuando se agregue o elimine una tableta. Las actualizaciones recientes a una tableta se almacenan en un buffer llamado memtable y las más viejas en SSTables, que son el formato en el que se almacenan datos en Bigtable. Estas se pueden usar para recuperar una tableta.

Comente los tipos de fallas de sistemas distribuidos en bases de datos que se mencionan en la lectura.

Hubo problemas de corrupción de memoria y red; errores fail-stop, que son cuando un componente para de funcionar sin tener otro comportamiento erróneo (GeeksforGeeks, 2023); error de un retraso grande en la propagación de señales; máquinas pegadas o muertas; particiones de red extendidas y asimétricas, que son cuando un sistema se divide en múltiples subgrupos, donde cada grupo no se puede comunicar con otros, solo los miembros de uno se pueden comunicar entre sí (Hazelcast, s.f.); y mantenimiento del hardware. Se pueden tomar varias medidas para disminuir los efectos de estas fallas. Se puede añadir un checksum para validar que los datos sean los correctos al comparar los checksums. No se debe asumir que otro sistema va a

funcionar, hay que implementar el manejo de errores de esos sistemas. Se puede tener un sistema de monitoreo para verificar que todo esté funcionando bien. Es mejor diseñar sistemas simples.

Bibliografía

Chang, F. W., Dean, J. M., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., & Gruber, R. (2006). BigTable: a distributed storage system for structured data. Operating Systems Design and Implementation, 205-218. https://doi.org/10.5555/1298455.1298475 GeeksforGeeks. (2023). Fail Stop failure in system design. GeeksforGeeks. https://www.geeksforgeeks.org/fail-stop-failure-in-system-design/ Network partitioning. (s. f.). https://docs.hazelcast.com/imdg/4.2/network-partitioning/network-partitioning