Bases de Datos I
16/10/2020
Resumen Cap 23 y 24 16/10
IS-501
José Inestroza

Capítulo 23 Fundamentals

Conceptos de bases de datos distribuidas

Las bases de datos distribuidas aportan las ventajas de la informática distribuida al dominio de la base de datos. Un sistema de computación distribuida consta de varios sitios o nodos de procesamiento que están interconectados por una red de computadoras y que cooperan en la realización de ciertas tareas asignadas. Como objetivo general, los sistemas informáticos distribuidos dividen un problema grande e inmanejable en piezas más pequeñas y lo resuelven de manera eficiente de manera coordinada. Por lo tanto, se aprovecha más potencia de cálculo para resolver una tarea compleja, y los nodos de procesamiento autónomos se pueden administrar de forma independiente mientras cooperan para proporcionar las funcionalidades necesarias para resolver el problema. La tecnología DDB resultó de la fusión de dos tecnologías: tecnología de bases de datos y tecnología de sistemas distribuidos. En las décadas de 1980 y 1990 se desarrollaron varios prototipos de sistemas de bases de datos distribuidas para abordar los problemas de distribución de datos, replicación de datos, procesamiento distribuido de consultas y transacciones, gestión de metadatos de bases de datos distribuidas y otros temas. Los orígenes de las tecnologías de big data provienen de sistemas distribuidos y sistemas de bases de datos, así como de algoritmos de minería de datos y aprendizaje automático que pueden procesar estas grandes cantidades de datos para extraer el conocimiento necesario.

Los sistemas NOSQL, que se enfocan en brindar soluciones distribuidas para administrar la gran cantidad de datos que se necesitan en aplicaciones como las redes sociales, la atención médica y la seguridad, por nombrar algunas. Los conceptos y sistemas que se utilizan para el procesamiento y análisis de big data, como map-reduce y otras tecnologías de procesamiento distribuido.

23.1 Conceptos de bases de datos distribuidas

Podemos definir una base de datos distribuida (DDB) como una colección de múltiples bases de datos interrelacionadas lógicamente distribuidas a través de una red informática, y un sistema de administración de bases de datos distribuidas (DDBMS) como un sistema de software que administra una base de datos distribuida mientras hace que la distribución sea transparente para el usuario.

23.1.1 Qué constituye una DDB

Para que una base de datos se llame distribuida, deben cumplirse las siguientes condiciones mínimas:

1. Conexión de nodos de bases de datos a través de una red informática.
2. Interrelación lógica de las bases de datos conectadas.
3. Posible ausencia de homogeneidad entre los nodos conectados. No es necesario que todos los nodos sean idénticos en términos de datos, hardware y software.

Todos los sitios pueden estar ubicados en proximidad física, por ejemplo, dentro del mismo edificio o un grupo de edificios adyacentes, y conectados a través de una red de área local, o pueden estar distribuidos geográficamente a grandes distancias y conectados a través de un área amplia o de largo recorrido. Las redes de área local suelen utilizar concentradores o cables inalámbricos, mientras que las redes de larga distancia utilizan líneas telefónicas, cables, infraestructuras de comunicación inalámbrica o satélites. Es común tener una combinación de varios tipos de redes. Las redes pueden tener diferentes topologías que definen las rutas de comunicación directa entre sitios.

23.1.2 Transparencia

El concepto de transparencia amplía la idea general de ocultar los detalles de implementación a los usuarios finales. Un sistema altamente transparente ofrece mucha flexibilidad al usuario final / desarrollador de aplicaciones, ya que requiere poca o ninguna conciencia de los detalles subyacentes de su parte. En el caso de una base de datos centralizada tradicional, la transparencia simplemente se refiere a la independencia lógica y física de los datos para los desarrolladores de aplicaciones. Son posibles los siguientes tipos de transparencias:

- Transparencia de la organización de datos (también conocida como transparencia de distribución o de red). Esto se refiere a la libertad para el usuario de los detalles operativos de la red y la ubicación de los datos en el sistema distribuido. Puede dividirse en transparencia de ubicación y transparencia de nomenclatura. La transparencia de ubicación se refiere al hecho de que el comando utilizado para realizar una tarea es independiente de la ubicación de los datos y la ubicación del nodo donde se emitió el comando. La transparencia de nombres implica que una vez que un nombre se asocia con un objeto, se puede acceder a los objetos nombrados sin ambigüedades sin especificación adicional en cuanto a dónde se encuentran los datos.
- Transparencia de replicación. Las copias de los mismos objetos de datos pueden almacenarse en varios sitios para mejorar la disponibilidad, el rendimiento y la confiabilidad. La transparencia de la replicación hace que el usuario desconozca la existencia de estas copias.
- Transparencia de fragmentación. Son posibles dos tipos de fragmentación. La fragmentación horizontal distribuye una relación (tabla) en subrelaciones que son subconjuntos de las tuplas (filas) en la relación original; esto también se conoce como fragmentación en los sistemas de computación en la nube y big data más nuevos. La fragmentación vertical distribuye una relación en subrelaciones donde cada subrelación está definida por un subconjunto de las columnas de la relación original. La transparencia de la fragmentación hace que el usuario desconozca la existencia de fragmentos.
- Otras transparencias incluyen transparencia de diseño y transparencia de ejecución, que se refieren, respectivamente, a la libertad de saber cómo está diseñada la base de datos distribuida y dónde se ejecuta una transacción.

23.1.3 Disponibilidad y confiabilidad

La confiabilidad y la disponibilidad son dos de las ventajas potenciales más comunes citadas para las bases de datos distribuidas. La confiabilidad se define ampliamente como la probabilidad de que un sistema esté funcionando (no inactivo) en un momento determinado, mientras que la disponibilidad es la probabilidad de que el sistema esté disponible continuamente durante un intervalo de tiempo. Podemos relacionar directamente la confiabilidad y disponibilidad de la base de datos con las fallas, errores y fallas asociadas con ella. Una falla puede describirse como una desviación del comportamiento de un sistema de lo especificado para asegurar la correcta ejecución de las operaciones. Los errores constituyen ese subconjunto de estados del sistema que causa la falla. La falla es la causa de un error.

Un enfoque común enfatiza la tolerancia a fallas; reconoce que ocurrirán fallas y diseña mecanismos que pueden detectar y eliminar fallas antes de que puedan resultar en una falla del sistema. Otro enfoque más estricto intenta garantizar que el sistema final no contenga fallas. Un DDBMS confiable tolera fallas de los componentes subyacentes y procesa las solicitudes de los usuarios siempre que no se viole la consistencia de la base de datos. Un administrador de recuperación de DDBMS tiene que lidiar con fallas que surgen de transacciones, hardware y redes de comunicación.

23.1.4 Escalabilidad y tolerancia de partición

La escalabilidad determina la medida en que el sistema puede expandir su capacidad mientras continúa funcionando sin interrupciones. Hay dos tipos de escalabilidad:
1. Escalabilidad horizontal: se refiere a expandir el número de nodos en el sistema distribuido.
2. Escalabilidad vertical: se refiere a expandir la capacidad de los nodos individuales en el sistema, como expandir la capacidad de almacenamiento o la potencia de procesamiento de un nodo.

El concepto de tolerancia de partición establece que el sistema debe tener la capacidad de continuar funcionando mientras la red está particionada.

23.1.5 Autonomía

La autonomía determina la medida en que los nodos o bases de datos individuales en una DDB conectada pueden operar de forma independiente. La autonomía se r al diseño, la comunicación y la ejecución. La autonomía de diseño se refiere a la independencia del uso del modelo de datos y las técnicas de gestión de transacciones entre los nodos. La autonomía de la comunicación determina la medida en que cada nodo puede decidir compartir información con otros nodos. La autonomía de ejecución se refiere a la independencia de los usuarios para actuar como les plazca.

23.1.6 Ventajas de las bases de datos distribuidas

Algunas ventajas importantes de DDB se enumeran a continuación.
1. Mayor facilidad y flexibilidad del desarrollo de aplicaciones. Desarrollo y mantenimiento de aplicaciones en sitios distribuidos geográficamente de una

organización se facilita debido a la transparencia de la distribución de datos y controlar.
2. Mayor disponibilidad. Esto se logra mediante el aislamiento de fallas en su sitio de origen sin afectar a los otros nodos de la base de datos conectados a la red. Se logran mejoras adicionales al replicar con criterio los datos y el software en más de un sitio. En un sistema centralizado, la falla en un solo sitio hace que todo el sistema no esté disponible para todos los usuarios. En una base de datos distribuida, algunos de los datos pueden ser inalcanzables, pero los usuarios aún pueden acceder a otras partes de la base de datos. Si los datos del sitio fallido se han replicado en otro sitio antes del fallo, el usuario no se verá afectado en absoluto. La capacidad del sistema para sobrevivir a la partición de la red también contribuye a la alta disponibilidad.

3. Desempeño mejorado. Un DBMS distribuido fragmenta la base de datos manteniendo los datos más cerca de donde más se necesitan. La localización de datos reduce la disputa por los servicios de CPU y E / S y simultáneamente reduce los retrasos de acceso involucrados en las redes de área amplia. Cuando una base de datos grande se distribuye en varios sitios, existen bases de datos más pequeñas en cada sitio. Como resultado, las consultas y transacciones locales que acceden a datos en un solo sitio tienen un mejor rendimiento debido a las bases de datos locales más pequeñas. Además, el paralelismo entre consultas e interconsultas se puede lograr ejecutando varias consultas en diferentes sitios o dividiendo una consulta en varias subconsultas que se ejecutan en paralelo. Esto contribuye a mejorar el rendimiento.

4. Expansión más sencilla mediante escalabilidad. En un entorno distribuido, la expansión del sistema en términos de agregar más datos, aumentar el tamaño de la base de datos o agregar más nodos es mucho más fácil que en los sistemas centralizados (no distribuidos).

La transparencia total proporciona al usuario global una vista de todo el DDBS como si fuera un único sistema centralizado. La transparencia se proporciona como complemento a la autonomía, lo que otorga a los usuarios un control más estricto sobre las bases de datos locales. Las características de transparencia pueden implementarse como parte del idioma del usuario, lo que puede traducir los servicios requeridos en operaciones apropiadas.

Capítulo 24 Fundamentals

Bases de datos NOSQL y sistemas de almacenamiento de Big Data

La clase de sistemas desarrollados para administrar grandes cantidades de datos en organizaciones como Google, Amazon, Facebook y Twitter y en aplicaciones como redes sociales, enlaces web, perfiles de usuario, marketing y ventas, publicaciones y tweets. , mapas de carreteras y datos espaciales, y correo electrónico. El término NOSQL es generalmente se interpreta como No solo SQL, en lugar de NO a SQL, y pretende transmitir que muchas aplicaciones necesitan sistemas distintos de los sistemas SQL relacionales tradicionales para aumentar sus necesidades de gestión de datos. La mayoría de los sistemas NOSQL son bases de datos distribuidas o sistemas de almacenamiento distribuidos, con un enfoque en el almacenamiento de datos semiestructurados, alto rendimiento, disponibilidad, replicación de datos y escalabilidad en lugar de un énfasis en la

coherencia de datos inmediata, potentes lenguajes de consulta y almacenamiento de datos estructurados.

24.1 Introducción a los sistemas NOSQL

24.1.1 Aparición de sistemas NOSQL

Muchas empresas y organizaciones se enfrentan a aplicaciones que almacenan grandes cantidades de datos. Considere una aplicación de correo electrónico gratuita, como Google Mail o Yahoo Mail u otro servicio similar; Existe la necesidad de un sistema de almacenamiento que pueda administrar todos estos correos electrónicos; un sistema SQL relacional estructurado puede no ser apropiado porque (1) los sistemas SQL ofrecen demasiados servicios (lenguaje de consulta potente, control de concurrencia, etc.) que esta aplicación puede no necesitar; y (2) un modelo de datos estructurado como el modelo relacional tradicional puede ser demasiado restrictivo. Algunas de las organizaciones que se enfrentaron a estas aplicaciones de gestión y almacenamiento de datos decidieron desarrollar sus propios sistemas:

- Google desarrolló un sistema NOSQL patentado conocido como BigTable, que se utiliza en muchas de las aplicaciones de Google que requieren una gran cantidad de almacenamiento de datos, como Gmail, Google Maps e indexación de sitios web. Apache Hbase es un sistema NOSQL de código abierto basado en conceptos similares. La innovación de Google llevó a la categoría de sistemas NOSQL conocidos como almacenes de columnas anchas o basadasen columnas; a veces también se les conoce como tiendas de familia de columnas.
- Amazon desarrolló un sistema NOSQL llamado DynamoDB que está disponible a través de los servicios en la nube de Amazon. Esta innovación llevó a la categoría conocida como almacenes de datos de valor clave o, a veces, almacenes de datos de tuplas clave o de objetos clave.
- Facebook desarrolló un sistema NOSQL llamado Cassandra, que ahora es de código abierto y se conoce como Apache Cassandra. Este sistema NOSQL utiliza conceptos tanto de almacenes de valores clave como de sistemas basados en columnas.
- Otras compañías de software comenzaron a desarrollar sus propias soluciones y las pusieron a disposición de los usuarios que necesitan estas capacidades, por ejemplo, MongoDB y CouchDB, que se clasifican como sistemas NOSQL basados en documentos o almacenes de documentos.
- Otra categoría de sistemas NOSQL son los sistemas NOSQL basados en gráficos o bases de datos de gráficos; estos incluyen Neo4J y GraphBase, entre otros.

24.1.2 Características de los sistemas NOSQL

Ahora discutimos las características de muchos sistemas NOSQL y cómo estos sistemas se diferencian de los sistemas SQL tradicionales. Dividimos las características en dos categorías: aquellas relacionadas con bases de datos distribuidas y sistemas distribuidos, y aquellas relacionadas con modelos de datos y lenguajes de consulta.

Características de NOSQL relacionadas con bases de datos distribuidas y sistemas distribuidos. Los sistemas NOSQL enfatizan la alta disponibilidad, por lo que replicar los datos es inherente a muchos de estos sistemas. La escalabilidad es otra característica importante, porque muchas de las aplicaciones que utilizan sistemas NOSQL tienden a tener datos que siguen creciendo en volumen. El alto

rendimiento es otra característica necesaria, mientras que la consistencia serializable puede no ser tan importante para algunas de las aplicaciones NOSQL. Algunas de estas características:
1. Escalabilidad Hay dos tipos de escalabilidad en los sistemas distribuidos: horizontal y vertical. En los sistemas NOSQL, generalmente se usa la escalabilidad horizontal, donde el sistema distribuido se expande agregando más nodos para el almacenamiento y procesamiento de datos a medida que crece el volumen de datos. La escalabilidad vertical, por otro lado, se refiere a expandir la capacidad de almacenamiento y computación de los nodos existentes.

2. Disponibilidad, replicación y consistencia eventual: muchas aplicaciones que utilizan sistemas NOSQL requieren una disponibilidad continua del sistema. Para lograr esto, los datos se replican en dos o más nodos de manera transparente, de modo que si un nodo falla, los datos aún están disponibles en otros nodos. La replicación mejora la disponibilidad de datos y también puede mejorar el rendimiento de lectura, porque las solicitudes de lectura a menudo se pueden atender desde cualquiera de los nodos de datos replicados. Muchas aplicaciones NOSQL no requieren consistencia serializable, por lo que formas más relajadas de consistencia conocida como eventual se utilizan consistencia.

3. Modelos de replicación: En los sistemas NOSQL se utilizan dos modelos de replicación principales: replicación maestro-esclavo y maestro-maestro. La replicación maestro-esclavo requiere que una copia sea la copia maestra; todas las operaciones de escritura deben aplicarse a la copia maestra y luego propagarse a las copias esclavas, usualmente usando consistencia eventual (las copias esclavas eventualmente serán las mismas que la copia maestra). Para lectura, el paradigma maestro-esclavo se puede configurar de varias formas. Una configuración requiere que todas las lecturas también estén en la copia maestra, por lo que esto sería similar al sitio principal o los métodos de copia principal de control de concurrencia distribuida con ventajas y desventajas similares. La replicación maestro-maestro permite lecturas y escrituras en cualquiera de las réplicas, pero es posible que no garantice que las lecturas en nodos que almacenan copias diferentes vean los mismos valores. Diferentes usuarios pueden escribir el mismo elemento de datos simultáneamente en diferentes nodos del sistema, por lo que los valores del elemento serán temporalmente inconsistentes.

4. Fragmentación de archivos: en muchas aplicaciones NOSQL, los archivos (o colecciones de objetos de datos) pueden tener muchos millones de registros (o documentos u objetos), y miles de usuarios pueden acceder a estos registros simultáneamente. Por lo tanto, no es práctico almacenar todo el archivo en un nodo. La fragmentación (también conocida como partición horizontal) de los registros de archivos se emplea a menudo en los sistemas NOSQL. Esto sirve para distribuir la carga de acceder a los registros del archivo a varios nodos. De los registros de archivos se emplea a menudo en los sistemas NOSQL. Esto sirve para distribuir la carga de acceder a los registros del archivo a varios nodos.

5. Acceso a datos de alto rendimiento: en muchas aplicaciones NOSQL, es necesario encontrar registros u objetos individuales (elementos de datos) entre los millones de registros de datos u objetos en un archivo. Para lograr esto, la mayoría de los sistemas utilizan una de dos técnicas: hash o partición de rango en claves de objeto. La mayoría de los accesos a un objeto serán proporcionando el valor clave en lugar de utilizar condiciones de consulta complejas. La clave de objeto es

similar al concepto de identificación de objeto En el hash, se aplica una función hash h (K) a la clave K, y la ubicación del objeto con la clave K está determinada por el valor de h (K). En la partición de rango, la ubicación se determina mediante un rango de valores clave; También se pueden usar otros índices para ubicar objetos basados en condiciones de atributo diferentes de la clave K.

Características de NOSQL relacionadas con modelos de datos y lenguajes de consulta. Los sistemas NOSQL enfatizan el rendimiento y la flexibilidad sobre el poder de modelado y las consultas complejas. A continuación, discutimos algunas de estas características.

1. No requerir un esquema: La flexibilidad de no requerir un esquema es Logrado en muchos sistemas NOSQL al permitir datos semiestructurados y auto descriptivos Los usuarios pueden especificar un esquema parcial en algunos sistemas para mejorar la eficiencia del almacenamiento, pero no es necesario tener un esquema en la mayoría de los sistemas NOSQL. Como puede que no haya un esquema para especificar restricciones, cualquier restricción sobre los datos debería programarse en los programas de aplicación que acceden a los elementos de datos. Hay varios lenguajes para describir datos semiestructurados, como JSON (notación de objetos JavaScript) y XML (lenguaje de marcado extensible).

2. Lenguajes de consulta menos potentes: muchas aplicaciones que utilizan sistemas NOSQL pueden no requerir un lenguaje de consulta potente como SQL, porque las consultas de búsqueda (lectura) en estos sistemas a menudo localizan objetos individuales en un único archivo en función de sus claves de objeto. Los sistemas NOSQL normalmente proporcionan un conjunto de funciones y operaciones como una API de programación (interfaz de programación de aplicaciones), por lo que la lectura y escritura de los objetos de datos se logra llamando a las operaciones apropiadas por parte del programador. En muchos casos, las operaciones se denominan operaciones CRUD, para Crear, Leer, Actualizar y Eliminar. En otros casos, se conocen como SCRUD debido a una operación de búsqueda (o búsqueda) agregada.

3. Control de versiones: algunos sistemas NOSQL proporcionan almacenamiento de múltiples versiones de los elementos de datos, con las marcas de tiempo de cuando se creó la versión de los datos.

24.1.3 Categorías de sistemas NOSQL

Los sistemas NOSQL se han caracterizado en cuatro categorías principales, con algunas categorías adicionales que abarcan otros tipos de sistemas. La categorización más común enumera las siguientes cuatro categorías principales:
1. Sistemas NOSQL basados en documentos: Estos sistemas almacenan datos en forma de documentos utilizando formatos conocidos, como JSON (JavaScript Object Notation). Se puede acceder a los documentos a través de su ID de documento, pero también se puede acceder rápidamente mediante otros índices.

2. Almacenes de clave-valor NOSQL: Estos sistemas tienen un modelo de datos simple basado en el acceso rápido de la clave al valor asociado con la clave; el valor puede ser un registro, un objeto o un documento o incluso tener una estructura de datos más compleja

3. Sistemas NOSQL basados en columnas o de columna ancha: Estos sistemas dividen una tabla por columna en familias de columnas, donde cada familia de columnas se almacena en sus propios archivos. También permiten el control de versiones de valores de datos.

4. Sistemas NOSQL basados en grafos: Los datos se representan como grafos y los nodos relacionados se pueden encontrar atravesando los bordes utilizando expresiones de ruta.

Se pueden agregar categorías adicionales de la siguiente manera para incluir algunos sistemas que no se clasifican fácilmente en las cuatro categorías anteriores, así como algunos otros tipos de sistemas que han estado disponibles incluso antes de que el término NOSQL se usara ampliamente.

5. Sistemas NOSQL híbridos: Estos sistemas tienen características de dos o más de las cuatro categorías anteriores.

6. Bases de datos de objetos.

7. Bases de datos XML

Incluso los motores de búsqueda basados en palabras clave almacenan grandes cantidades de datos con un acceso de búsqueda rápido, por lo que los datos almacenados pueden considerarse como grandes almacenes de datos grandes de NOSQL.