Sistemas Distribuidos

Bloque IV

Sistemas de persistencia distribuidos

Tema 5

Bases de Datos Distribuidas

Nicolás Rodríguez
nicolas.rodriguez@urjc.es

Índice

- **Sistemas de persistencia distribuidos**
- Arquitecturas
- Centralización vs. Distribución
- Tecnologías y herramientas

Sistemas de persistencia distribuidos

- Es un tipo de sistema de almacenamiento de datos que distribuye y gestiona la información a través de múltiples ubicaciones físicas o lógicas.
- Los datos no se almacenan en una única ubicación central, sino repartidos en varios nodos.
- Principios fundamentales:
  - Distribución de datos: Los datos se almacenan en múltiples nodos, lo que aumenta la redundancia y la resistencia a fallos.
  - Independencia de ubicación: Los datos pueden ser accesibles desde cualquier nodo, mejorando el acceso y la disponibilidad.
  - Escalabilidad: La capacidad de agregar más nodos para incrementar el almacenamiento y el rendimiento sin interrumpir el servicio existente.

Sistemas de persistencia distribuidos

- Nodos de almacenamiento:
  - Cada nodo puede ser un servidor, un PC o un dispositivo de almacenamiento dedicado.
  - Cada nodo puede operar de manera independiente, pero también se coordina con otros nodos.
- Red de comunicaciones:
  - Red que conecta los nodos, permitiendo el intercambio de datos y mensajes de control.
  - La eficiencia y la seguridad de la red impactan en el rendimiento del sistema.
- Software de gestión:
  - Software que administra la distribución de datos, la consistencia, y la recuperación en caso de fallos.
  - Ejemplos de operaciones gestionadas: replicación de datos, balanceo de carga, y manejo de errores.

Sistemas de persistencia distribuidos

- Datos estructurados:
  - Bases de datos relacionales, donde los datos se organizan en tablas y columnas.
- Datos no estructurados:
  - Archivos multimedia, documentos, y registros de logs que no siguen un modelo estructurado.
- Datos semiestructurados:
  - Datos que no están organizados en un formato rígido como el relacional, pero que contienen etiquetas o marcas que permiten identificar elementos de los datos, como XML y JSON.

Sistemas de persistencia distribuidos

- Estos sistemas subyacen en servicios comunes como:
  - Almacenamiento en la nube.
  - Plataformas de streaming.
  - Redes sociales.
- Permiten gestionar y acceder a enormes cantidades de datos de manera eficiente.
- Importante en áreas como:
  - Salud (registros médicos electrónicos).
  - Finanzas (transacciones y análisis de mercado en tiempo real).
  - Comercio electrónico (gestión de inventarios y procesamiento de pedidos).

Sistemas de persistencia distribuidos

- Escalabilidad y flexibilidad:
  - Los SSDD pueden manejar incrementos en la demanda de datos y tráfico de usuarios más eficientemente que los sistemas centralizados, adaptándose a las necesidades cambiantes sin interrupciones significativas.
- Resiliencia y disponibilidad:
  - Estos sistemas mantienen la operatividad incluso ante fallos de hardware o software en ciertos nodos, asegurando así una mayor continuidad del servicio.
- Eficiencia en el procesamiento y acceso a datos:
  - La distribución de datos y carga de trabajo facilita un acceso más rápido y eficiente a los datos, crucial para aplicaciones que requieren tiempos de respuesta bajos.

Índice

- Sistemas de persistencia distribuidos
- **Arquitecturas:**
  - P2P
  - Cliente-Servidor
  - Fragmentación
  - Federada
- Centralización vs. Distribución
- Tecnologías y herramientas

Arquitecturas: P2P

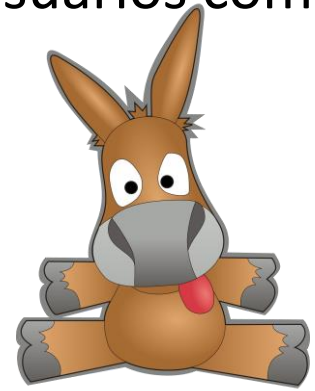
- Descentralización:
  - En los sistemas P2P, no existe un servidor centralizado o nodo administrador.
  - Cada nodo (o "par") en la red tiene capacidades similares y puede actuar tanto como cliente como servidor.
  - Esta descentralización conlleva una distribución equitativa de las responsabilidades de almacenamiento, procesamiento y transmisión de datos entre todos los nodos.
- Auto-organización:
  - Los nodos en una red P2P se organizan y coordinan automáticamente para compartir recursos y datos.
  - Esta capacidad de auto-organización hace que los sistemas P2P sean robustos y flexibles frente a cambios dinámicos en la red, como la adición o eliminación de nodos.

Arquitecturas: P2P

- Descubrimiento y conexión de Pares:
  - Los nodos utilizan mecanismos de descubrimiento para encontrar otros nodos y establecer conexiones. Esto puede incluir directorios centralizados en redes P2P híbridas o algoritmos descentralizados en redes puramente P2P.
- Intercambio y replicación de datos:
  - Los datos se comparten directamente entre nodos sin pasar por un servidor central (tanto la transmisión de archivos como la distribución de fragmentos de datos).
  - La replicación de datos en varios nodos mejora la disponibilidad y la resistencia a fallos.
- Balance de carga y redundancia:
  - En una red P2P, la carga se distribuye de manera más uniforme entre los nodos, evitando cuellos de botella.
  - La redundancia inherente en estos sistemas asegura la disponibilidad continua de los datos, incluso si algunos nodos fallan.

Arquitecturas: P2P

- Redes de intercambio de archivos:
  - Las redes P2P son conocidas por su uso en el intercambio de archivos, donde los usuarios comparten archivos de manera directa sin la necesidad de un servidor central.



- Sistemas de comunicación:
  - Aplicaciones de mensajería y llamadas que utilizan una infraestructura P2P para una comunicación directa y eficiente entre usuarios.
- Blockchains y criptomonedas:
  - Las tecnologías blockchain, como Bitcoin, utilizan redes P2P para mantener un registro distribuido y seguro de transacciones.

Arquitecturas: P2P

- Ventajas:
  - Resistencia a fallos y censura debido a la falta de un punto central de control o fallo.
  - Escalabilidad, ya que añadir más nodos a menudo aumenta la capacidad y el rendimiento de la red.

- Desafíos:
  - Dificultades en garantizar la seguridad y la privacidad, dada la naturaleza abierta y descentralizada de la red.
  - Problemas de consistencia de datos, especialmente en redes grandes y altamente dinámicas.

Arquitecturas: Cliente – Servidor

- Estructura de rol definido:
  - En esta arquitectura, los roles están claramente definidos: los servidores proporcionan recursos o servicios, mientras que los clientes solicitan y utilizan estos servicios.
  - A diferencia de los sistemas P2P, hay una distinción clara entre los nodos que ofrecen servicios (servidores) y los que los consumen (clientes).
- Distribución de servidores:
  - Los servidores en este modelo están distribuidos geográficamente o en diferentes sistemas para mejorar la escalabilidad y la disponibilidad.
  - Esta distribución puede ser transparente para los clientes, quienes interactúan con el sistema como si fuera un único servidor.

Arquitecturas: Cliente – Servidor

- Balanceo de carga:
  - Los SSDD cliente-servidor a menudo implementan mecanismos de balanceo de carga para distribuir las solicitudes de los clientes de manera uniforme entre múltiples servidores, evitando así la sobrecarga de cualquier servidor individual.
- Replicación y redundancia de datos:
  - Los datos importantes pueden ser replicados en múltiples servidores, lo que garantiza que la información esté disponible incluso si uno de los servidores falla.
  - La replicación puede ser manejada a nivel de aplicación o mediante infraestructura de almacenamiento distribuido.
- Gestión de sesiones:
  - En entornos donde los clientes interactúan con el sistema en sesiones (como en aplicaciones web), la arquitectura debe manejar la persistencia y consistencia de la sesión a través de múltiples servidores.

Arquitecturas: Cliente – Servidor

- Aplicaciones web y móviles:
  - Las aplicaciones web y móviles a menudo utilizan una arquitectura cliente-servidor distribuida para servir contenido y servicios a una gran base de usuarios, distribuyendo la carga entre múltiples servidores de back-end.

- Juegos en línea y servicios de streaming:
  - Los juegos en línea y las plataformas de streaming utilizan servidores distribuidos para proporcionar una experiencia de usuario fluida y de baja latencia, manejar grandes volúmenes de tráfico y datos en tiempo real.

Arquitecturas: Cliente – Servidor

- Ventajas:
  - Mejor escalabilidad y capacidad de manejar grandes volúmenes de solicitudes y datos.
  - Mayor disponibilidad y redundancia, lo que reduce el riesgo de tiempo de inactividad.

- Desafíos:
  - Mayor complejidad en la gestión y mantenimiento de la infraestructura distribuida.
  - Necesidad de mecanismos efectivos para la sincronización y coherencia de datos entre servidores.

Arquitecturas: Fragmentación

- En un modelo basado en shards, la base de datos se divide en piezas más pequeñas y manejables, conocidas como "shards" o fragmentos.
- Cada shard contiene una porción del total de datos.
- Esta división se realiza para mejorar el rendimiento, la escalabilidad y la gestión de los datos.
- Distribución de shards:
  - Los shards se distribuyen a través de múltiples nodos o servidores, lo que permite que las operaciones de datos se realicen paralelamente en diferentes shards.
  - La distribución puede basarse en diversos criterios, como rango de claves, hash de claves o incluso de manera manual.

Arquitecturas: Fragmentación

- Clave de shard y enrutamiento:
  - Cada shard está asociado con una clave o un rango de claves. Cuando se realiza una solicitud de datos, un mecanismo de enrutamiento determina a cuál shard dirigir la consulta basándose en esta clave.

- Equilibrio y replicación de shards:
  - Para evitar desequilibrios en la carga de trabajo entre los shards, se implementan estrategias de equilibrio que redistribuyen los datos cuando es necesario.
  - Los shards suelen replicarse en múltiples nodos para garantizar la disponibilidad y la resistencia a fallos.

Arquitecturas: Fragmentación

- BBDD de gran volumen:
  - Los modelos basados en shard son especialmente útiles en bases de datos que manejan grandes volúmenes de datos y requieren operaciones de lectura y escritura de alta velocidad (MongoDB).
  - Ejemplos: BBDD y RRSS
- Sistemas de análisis de datos en tiempo real:
  - Aplicaciones que requieren análisis y procesamiento rápido de datos, como sistemas de monitoreo y análisis en tiempo real, se benefician del paralelismo que ofrecen los shards.

Arquitecturas: Fragmentación

- Ventajas:
  - Mayor escalabilidad, ya que agregar más nodos permite distribuir los shards de manera más amplia y manejar más datos.
  - Mejora en el rendimiento, debido al procesamiento paralelo y la reducción de cuellos de botella.

- Desafíos:
  - Complejidad en el diseño y la gestión, especialmente en cuanto a la distribución equitativa de los datos y la gestión de shards.
  - Dificultades en mantener la consistencia y realizar transacciones complejas a través de múltiples shards.

Arquitecturas: Federadas

- Una arquitectura federada conecta diferentes sistemas de BBDD o sistemas de almacenamiento, cada uno con su propia gestión y almacenamiento de datos, en un único sistema federado.

- Permite a los usuarios acceder y manipular datos en múltiples bases de datos como si fueran una sola entidad.
- Autonomía y heterogeneidad:
  - Cada sistema en una federación mantiene cierto grado de autonomía, operando bajo su propia administración y con sus propios esquemas de datos.
  - La arquitectura es capaz de manejar la heterogeneidad, integrando sistemas que pueden diferir en términos de modelos de datos, plataformas y tecnologías.

Arquitecturas: Federadas

- Interfaz de acceso unificado:
  - La arquitectura federada proporciona una interfaz unificada para acceder a los datos, lo que simplifica las consultas y transacciones que abarcan múltiples bases de datos.
- Traducción y mapeo de esquemas:
  - Se utilizan mecanismos para traducir y mapear diferentes esquemas de datos, permitiendo que las consultas se realicen de manera coherente a través de sistemas heterogéneos.
- Gestión de consultas y transacciones:
  - La arquitectura debe manejar eficientemente las consultas y transacciones distribuidas, asegurando que los resultados sean consistentes y fiables a pesar de la distribución y autonomía de los sistemas individuales.

Arquitecturas: Federadas

- Entornos empresariales multidepartamentales:
  - Las organizaciones con diferentes departamentos que operan sistemas de datos separados pueden utilizar arquitecturas federadas para un acceso y análisis de datos integrados sin comprometer la autonomía departamental.

- Colaboración entre organizaciones:
  - Permite a distintas organizaciones colaborar y compartir datos sin necesidad de fusionar sus sistemas de almacenamiento de datos existentes.

Arquitecturas: Federadas

- Ventajas:
  - Flexibilidad y capacidad para integrar sistemas de datos diversos manteniendo la independencia de cada sistema.
  - Facilita el acceso y análisis de datos a gran escala, potenciando la toma de decisiones basada en información más completa y variada.

- Desafíos:
  - Complejidad en la coordinación y gestión de datos entre sistemas autónomos.
  - Problemas potenciales en el rendimiento debido a la sobrecarga generada por la integración de múltiples sistemas.

Índice

- Sistemas de persistencia distribuidos
- Arquitecturas
- **Centralización vs. Distribución**
- Tecnologías y herramientas

Centralización vs. Distribución

- Ubicación de los datos:
  - En sistemas centralizados, todos los datos se almacenan en un único servidor o ubicación central. Esto puede simplificar la gestión y el control, pero también crea un punto único de fallo y puede limitar la escalabilidad.
  - Los sistemas distribuidos almacenan datos en múltiples nodos, lo que reduce el riesgo de un punto único de fallo y permite una mayor escalabilidad y resistencia a fallos.
- Acceso a los datos:
  - Los sistemas centralizados pueden sufrir cuellos de botella en el rendimiento cuando hay múltiples solicitudes simultáneas, debido a la limitada capacidad de procesamiento del servidor central.
  - En los SSDD, las solicitudes se pueden manejar en paralelo a través de varios nodos, mejorando el rendimiento y la eficiencia.

Centralización vs. Distribución

- Complejidad administrativa:
  - Los sistemas centralizados suelen ser más fáciles de administrar debido a su estructura unificada.
  - La actualización, el mantenimiento y la seguridad pueden gestionarse de forma centralizada.
  - Los SSDD requieren una gestión más compleja debido a su naturaleza dispersa.
  - La coordinación entre nodos, la sincronización de datos y la seguridad distribuida presentan desafíos adicionales.
- Resiliencia y recuperación de desastres:
  - En sistemas centralizados, la recuperación de desastres puede ser más complicada, ya que un fallo en el servidor central puede impactar todo el sistema.
  - Los SSDD, con su redundancia de datos incorporada y la distribución geográfica, suelen ser más resistentes a fallos y desastres, facilitando la recuperación.

Centralización vs. Distribución

- Escalabilidad:
  - Los sistemas centralizados pueden tener dificultades para escalar, ya que esto a menudo requiere actualizaciones de hardware caras y complejas.
  - Los SSDD permiten la escalabilidad horizontal, agregando más nodos para aumentar la capacidad, lo cual es generalmente más económico y flexible.

- Rendimiento bajo alta demanda:
  - Los sistemas centralizados pueden sufrir degradaciones en el rendimiento bajo cargas de trabajo pesadas, mientras que los SSDD pueden manejar mejor estas demandas al distribuir la carga entre varios nodos.

Índice

- Sistemas de persistencia distribuidos
- Arquitecturas
- Centralización vs. Distribución
- **Tecnologías y herramientas**
  - BBDD relacionales distribuidas
  - BBDD NoSQL
  - Sistemas de archivos distribuidos

BBDD relacionales distribuidas

- Las BBDD distribuidas almacenan datos en varios nodos, que pueden estar ubicados en diferentes servidores o incluso en diferentes centros de datos.
- A pesar de la distribución física, estos sistemas presentan una vista unificada de los datos al usuario o a la aplicación.
- La replicación involucra mantener copias de datos en múltiples nodos para garantizar la disponibilidad y resistencia a fallos.
- El particionamiento distribuye los datos en diferentes nodos para mejorar el rendimiento y la escalabilidad.

BBDD relacionales distribuidas

- Las bases de datos relacionales distribuidas representan una extensión del modelo relacional tradicional para operar en un entorno distribuido.

- Estructura relacional en un entorno distribuido:
  - Mantienen la estructura relacional de tablas, filas y columnas, pero los datos se almacenan y procesan en múltiples nodos de una red.
  - A pesar de la distribución física, presentan una vista unificada de la base de datos al usuario.

BBDD relacionales distribuidas

- Coherencia de datos:
  - Al igual que las bases de datos relacionales tradicionales, enfatizan la integridad y la coherencia de los datos, manteniendo las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) a través de transacciones distribuidas.

- Replicación y particionamiento de datos:
  - Implementan la replicación de datos para garantizar la disponibilidad y la tolerancia a fallos.
  - El particionamiento de datos permite distribuir las cargas de trabajo de manera más eficiente a través de los nodos.

BBDD relacionales distribuidas

- Transacciones distribuidas:
  - Gestionan transacciones que involucran datos en múltiples ubicaciones, asegurando la coherencia y atomicidad a lo largo de toda la operación.
  - Utilizan protocolos como el compromiso de dos fases (Two-Phase Commit) para garantizar que todas las partes de una transacción se completen con éxito o se deshagan por completo.

- Consultas y acceso a datos:
  - Las consultas se ejecutan de manera que pueden acceder y combinar datos de diferentes nodos, proporcionando resultados como si los datos provinieran de una sola fuente.

BBDD relacionales distribuidas

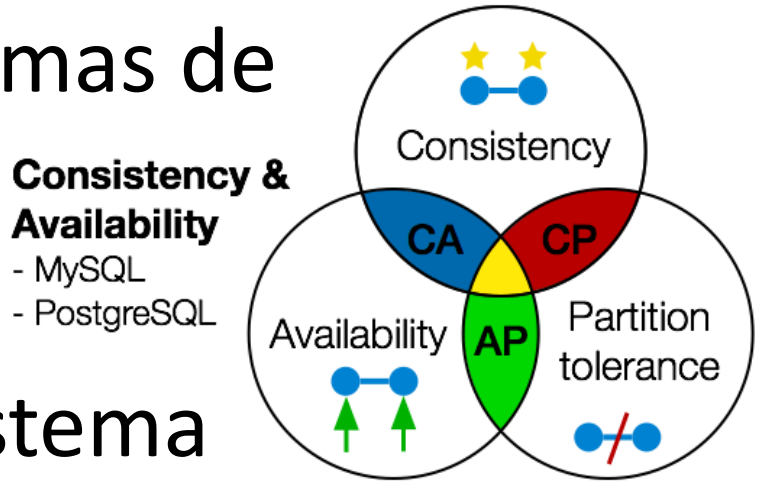
- Oracle Real Application Clusters (RAC):
  - Permite que múltiples instancias de la base de datos Oracle accedan y procesen un conjunto compartido de datos.
  - Es ampliamente utilizado en entornos empresariales para aplicaciones críticas que requieren alta disponibilidad y escalabilidad.

- Microsoft SQL Server Always On:
  - Una solución de alta disponibilidad y recuperación ante desastres que permite a los usuarios configurar múltiples réplicas de bases de datos SQL Server en diferentes nodos.
  - Proporciona un entorno operativo continuo y protección contra fallos.

Teorema CAP

- El teorema CAP, también conocido como Principio de Brewer, es un concepto fundamental en el diseño de sistemas de bases de datos distribuidas.

- Este teorema establece que un sistema de base de datos distribuida solo puede proporcionar dos de las siguientes tres garantías simultáneamente:



Consistency & Availability
- MySQL
- PostgreSQL

Consistency & Partition tolerance
- HBase
- MongoDB
- Redis
- Memcache

Availability & Partition tolerance
- Riak
- Cassandra
- CouchDB
- DynamoDB

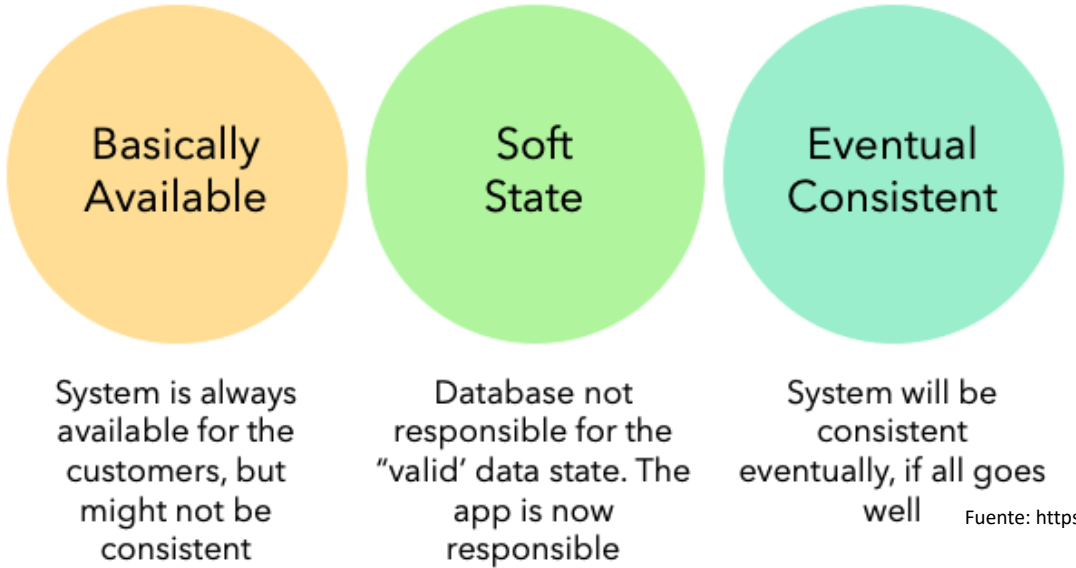
Fuente: https://debeando.com/teorema-cap.html

Teorema CAP

- Consistencia (C):
  - Todos los nodos ven los mismos datos al mismo tiempo. Cualquier lectura recibirá la versión más reciente de un dato escrito.
- Disponibilidad (A):
  - Cada solicitud recibe una respuesta sobre si fue exitosa o fallida, pero sin garantizar que todos los nodos contengan la misma versión más reciente del dato.
- Tolerancia a Particiones de Red (P):
  - El sistema continúa funcionando a pesar de cualquier número de fallos de comunicación entre los nodos.
- En la práctica, se tiene que elegir un par.

BASE

- BASE es un acrónimo que describe las propiedades de los sistemas que no siguen estrictamente las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) típicas de las bases de datos relacionales tradicionales.
- En cambio, BASE se enfoca en la escalabilidad y rendimiento en sistemas de bases de datos distribuidas.



Fuente: https://f5sal.medium.com/database-selection-design-part-iv-f4577ba049c5

BASE

- Basicamente Disponible (Basically Available):
  - Indica que el sistema garantiza la disponibilidad en términos de capacidad para realizar operaciones, aunque no garantiza una respuesta inmediata o coherente.
- Soft-state:
  - El estado del sistema puede cambiar con el tiempo, incluso sin entrada. Esto se debe a que el sistema puede eventualmente propagar actualizaciones a todos los nodos, pero no garantiza que ocurra inmediatamente.
- Consistencia Eventual (Eventual Consistency):
  - Aunque el sistema no garantiza la consistencia inmediata, asegura que, eventualmente, todos los nodos contendrán los mismos datos.

BBDD NoSQL

- BBDD Not only SQL (BBDD NoSQL).
- Son un conjunto diverso de tecnologías de bases de datos que se diseñaron para abordar varias limitaciones de las bases de datos relacionales tradicionales.
- Especialmente útiles en contextos de grandes volúmenes de datos, alta escalabilidad y operaciones distribuidas.
- A diferencia de las bases de datos relacionales, las BBDD NoSQL no se basan estrictamente en tablas y a menudo no usan SQL como su principal lenguaje de consulta.

BBDD NoSQL

- Esquemas flexibles:
  - No requieren un esquema fijo o definido previamente.
  - Esto permite almacenar datos que pueden tener estructuras diversas sin necesidad de modificar el diseño de la base de datos.
- Escalabilidad horizontal:
  - Están diseñadas para escalar fácilmente añadiendo más máquinas o nodos al sistema, lo que es ideal para aplicaciones que necesitan manejar un gran volumen de tráfico o datos.
- Modelos de datos diversos:
  - Soportan una variedad de modelos de datos, incluyendo clave-valor, documentos, columnas anchas y grafos, cada uno adecuado para diferentes tipos de aplicaciones y requisitos de uso.

BBDD NoSQL

- Coherencia eventual:
  - Muchas bases de datos NoSQL ofrecen una consistencia eventual, lo que significa que todas las copias de los datos eventualmente se sincronizarán, pero no necesariamente inmediatamente después de una operación de escritura.

- Optimizadas para rendimiento y velocidad:
  - En general, están optimizadas para operaciones rápidas de lectura y escritura y pueden ser más eficientes que las bases de datos relacionales para ciertos tipos de consultas y operaciones.

BBDD NoSQL

- Clave-Valor:
  - Almacenan los datos en pares de clave-valor.
  - Son simples y eficientes para operaciones de lectura y escritura.
  - Ejemplo: Redis.

- Documentos:
  - Almacenan datos en documentos (generalmente en formatos como JSON o BSON), lo que los hace ideales para almacenar datos complejos y jerárquicos.
  - Ejemplo: MongoDB.

BBDD NoSQL

- Columnas anchas:
  - Optimizadas para consultas en grandes conjuntos de datos, almacenan datos en filas y columnas, pero de manera más flexible y distribuida que las BBDD relacionales. Ejemplo: Apache Cassandra.

- Grafos:
  - Diseñadas para almacenar y manejar datos relacionales complejos, son ideales para representar redes y relaciones.
  - Ejemplo: Neo4j.

Sistemas de archivos distribuidos

- Los sistemas de archivos distribuidos son tecnologías clave en los sistemas de persistencia distribuidos.

- Están diseñados para almacenar y acceder a archivos y datos a través de una red de nodos, facilitando el acceso y la manipulación de datos distribuidos como si estuvieran almacenados en una ubicación local.

Sistemas de archivos distribuidos

- Almacenamiento de datos en múltiples nodos:
  - En un sistema de archivos distribuido, los archivos y datos se almacenan en varios servidores o nodos en una red.
  - Esto permite que los datos se almacenen de manera eficiente y estén accesibles desde cualquier nodo del sistema.
- Transparencia de ubicación:
  - Para los usuarios y aplicaciones, el sistema parece ser un único sistema de archivos coherente, independientemente de dónde se almacenen físicamente los datos.
  - Esto es conocido como transparencia de ubicación.
- Escalabilidad y fiabilidad:
  - Estos sistemas están diseñados para ser altamente escalables, permitiendo añadir más nodos de almacenamiento para aumentar la capacidad.
  - También ofrecen redundancia y tolerancia a fallos, replicando datos en nodos.

Sistemas de archivos distribuidos

- Replicación y distribución de datos:
  - Los datos se replican en múltiples nodos para garantizar la disponibilidad y la resistencia a fallos.
  - La distribución de datos puede ser basada en varios factores, como la carga, la proximidad geográfica o la frecuencia de acceso.
- Consistencia y sincronización:
  - Los sistemas de archivos distribuidos utilizan diversos algoritmos para mantener la consistencia de los datos entre los nodos.
  - Estos pueden incluir modelos de coherencia eventual o sistemas más estrictos de coherencia.

Sistemas de archivos distribuidos

- Hadoop Distributed File System (HDFS):
  - Un componente clave del ecosistema Hadoop, HDFS es ideal para aplicaciones con grandes conjuntos de datos.
  - Distribuye los datos en bloques a través de múltiples nodos, proporcionando alta tolerancia a fallos y facilidad de escalabilidad.
- Google File System (GFS):
  - Diseñado para aplicaciones que requieren un procesamiento de datos extenso, GFS maneja el almacenamiento de datos en una gran cantidad de nodos, optimizando la eficiencia y la confiabilidad.
- Network File System (NFS):
  - Uno de los sistemas de archivos distribuidos más antiguos y comunes, NFS permite a los usuarios acceder a archivos en una red de manera similar a cómo accederían a los almacenados en su propio sistema.

Sistemas de archivos distribuidos

- Ventajas:
  - Escalabilidad para manejar grandes volúmenes de datos.
  - Alta disponibilidad de datos y resistencia a fallos a través de la replicación.
  - Acceso y gestión de archivos eficientes a través de redes.

- Desafíos:
  - Mantener la consistencia de los datos puede ser complejo, especialmente en sistemas con alta latencia de red o cuando se requiere coherencia en tiempo real.
  - La gestión y el mantenimiento de un sistema de archivos distribuido a gran escala pueden ser técnicamente desafiantes.

Sistemas Distribuidos

Bloque IV

Sistemas de persistencia distribuidos

Tema 5

Bases de Datos Distribuidas

URJC Escuela Técnica Superior de Ingeniería Informática

Nicolás Rodríguez
nicolas.rodriguez@urjc.es