Estructura del Sistema Distribuido

Arquitectura

La organización del sistema distribuido se basa en la arquitectura Chord, en esta organización:

- Cada nodo del sistema tiene un identificador único basado en un hash de su dirección.
- Los archivos se asocian con un identificador único derivado del contenido del archivo (hash del contenido). Este identificador se utiliza para asignar el archivo a un nodo específico en el anillo Chord.

Roles del sistema

En el sistema hay dos roles principales:

- Clientes:
  - Suben archivos al sistema. El cliente calcula el hash del contenido del archivo y lo envía al nodo correspondiente en el anillo Chord.
  - Buscan archivos por nombre o extensión. Estas consultas se resuelven mediante una exploración en el anillo para encontrar coincidencias en las tablas de metadatos de los nodos.
  - Descargan archivos desde los nodos correspondientes.
- Servidores (nodos del anillo Chord):
  - Almacenan los archivos y sus metadatos (incluyendo los nombres alternativos asociados a un mismo contenido).
  - Gestionan la pertenencia al anillo (unión y salida de nodos).
  - Procesan las solicitudes de clientes, como búsquedas, almacenamiento y recuperación de archivos.
  - Coordinar la replicación de los datos.

Distribución de servicios en redes Docker

El sistema se implementa en redes Docker separadas para clientes y servidores, con un router que las interconecta:

- Red de servidores:
  - Esta red aloja todos los nodos del anillo Chord.
  - Los nodos se comunican directamente entre ellos para mantener la estructura del anillo, realizar el enrutamiento de claves y almacenar los datos.

    - Los servicios de mantenimiento del anillo (por ejemplo, estabilización de nodos, actualización de fingers) se ejecutan exclusivamente aquí.
- Red de clientes:
    - Los clientes se encuentran en esta red y no tienen comunicación directa entre ellos.
    - Se conectan al router para interactuar con los nodos del anillo.
- Router:
    - Media la comunicación entre las redes de clientes y servidores.
    - Garantiza que las solicitudes de los clientes (como subir, buscar y descargar archivos) lleguen a los nodos correspondientes en la red de servidores.

Tipos de Procesos

Cada programa (cliente o servidor) contiene múltiples procesos y componentes, organizados de la siguiente manera:

Cliente

Solicitudes como "subir archivo", "buscar archivo", y "descargar archivo" generan eventos que se procesan en hilos.

Gestión de Comunicaciones:

Hilos dedicados a enviar y recibir mensajes desde y hacia los servidores.
Cada hilo puede manejar una solicitud (por ejemplo, enviar un archivo o recibir datos).

Procesamiento de Datos:

Hilos encargados de procesar los datos del usuario, como calcular hashes de archivos o interpretar los resultados de búsqueda.

Uso de hilos en el sistema

Cada servidor en el anillo Chord tiene una arquitectura modular y organiza sus hilos en grupos según las responsabilidades principales. A continuación, se describe la estructura detallada:

1. Hilo Principal

- Responsabilidad:
    - Inicia el nodo y los servicios del servidor.
    - Acepta nuevas conexiones de clientes o nodos.
    - Despacha las solicitudes entrantes a hilos secundarios para su procesamiento.

2. Grupo de Hilos para Solicitudes de Clientes

- Responsabilidad:
  - Manejar las solicitudes de subida, búsqueda y descarga de archivos de los clientes.
  - Procesar cada solicitud de manera independiente y devolver una respuesta.

3. Grupo de Hilos para Comunicaciones entre Nodos

- Responsabilidad:
  - Manejar las solicitudes de enrutamiento y redirección entre nodos del anillo.
  - Actualizar la tabla finger y otros datos necesarios para mantener la topología del anillo.

4. Hilos de Mantenimiento del Anillo

- Responsabilidad:
  - Garantizar que la estructura del anillo Chord sea consistente y funcional.
  - Realizar tareas periódicas como:
    - Verificar que los nodos sucesores y predecesores están activos.
    - Reparar el anillo si algún nodo falla.
    - Actualizar la tabla finger según cambios en el anillo.

Comunicación

El sistema usa un enfoque basado en mensajes para la comunicación entre los servidores y clientes, así como para la comunicación entre los servidores en sí.
Para esto utilizaremos ZMQ, la cual ofrece diferentes patrones de mensajería que permiten implementar todas las interacciones necesarias entre cliente-servidor, servidor-servidor, e interprocesos dentro de un nodo.

Comunicación Cliente-Servidor

Se utiliza el patrón Request-Reply de ZMQ, en el cuál el cliente puede realizar una solicitud al servidor, ya sea de subida, búsqueda o descarga, y el servidor podrá enviar una respuesta acorde a la solicitud.

Comunicación Servidor-Servidor

Se utiliza el patrón Request-Reply para que los nodos del servidor envien solicitudes al resto de nodos para que realicen operaciones como comprobar si un archivo pertenece a ellos.
Además se usa el patrón Publish-Subscribe para operaciones de mantenimiento del anillo Chord, ya que estas pueden permitir a los nodos conocer que componentes del anillo están activas.

Coordinaación

Para evitar que dos procesos intenten escribir a la vez en la base de datos de un nodo podemos usar un mecanismo de atención exclusiva, en el caso de python con un `threading.Lock` podemos garantizar que solo un proceso acceda a la base de datos a la vez.

Dado que en un anillo Chord los nodos son independientes, no es necesario que estos realicen operaciones conjuntas sincronizadas.

Nombrado y Localización

La arquitectura Chord ya define gran parte de los mecanismos necesarios para el nombrado y localización de recursos en un sistema distribuido. En Chord, los recursos se identifican mediante claves hash, y cada nodo en el anillo también tiene un identificador hash único. Cada archivo es identificado por un hash único calculado a partir de su contenido. Esto garantiza que dos archivos con el mismo contenido tengan el mismo identificador, independientemente de sus nombres.Los nombres adicionales de un archivo se almacenan junto con su identificador único en la base de datos del nodo responsable. Los nodos en el sistema (servidores) se identifican por su ID hash derivado de su dirección IP.

Consistencia y Replicación

En Chord, los datos se distribuyen de manera uniforme en los nodos del anillo, gracias al uso del hash. El nodo responsable de un dato (archivo) es aquel cuyo identificador es el sucesor inmediato del hash del archivo.

Replicación: Cantidad de Réplicas

La replicación asegura que los datos estén disponibles incluso si un nodo falla.

- Estrategia de replicación utilizada:
  - Cada archivo tiene (k) réplicas, almacenadas en los (k) sucesores más cercanos al nodo responsable.
    Al almacenar un archivo, el nodo responsable envía copias del archivo a sus (k) sucesores inmediatos.

Manejo de Fallos en las Réplicas

En sistemas distribuidos, las réplicas pueden volverse inaccesibles debido a fallos de nodos. Es crucial implementar un mecanismo para detectar y recuperar réplicas.

1. Detección de fallos:
   - Usa un sistema de heartbeats entre los nodos responsables y sus sucesores.

    - Si un sucesor no responde, el nodo responsable debe tomar medidas para replicar el archivo en el siguiente nodo disponible.
2. Recuperación de réplicas:
    - Cuando un nuevo nodo entra en el sistema, debe recibir las réplicas de los datos que le corresponden según su posición en el anillo.
    - Este proceso puede iniciarse automáticamente tras detectar cambios en la topología del anillo.

Tolerancia a fallas

Para mantener el sistema funcional en presencia de fallos, es necesario implementar mecanismos que detecten, gestionen y respondan a los errores:

Detección de fallos:

- Cada nodo envía mensajes periódicos a sus sucesores y predecesores inmediatos para confirmar que están activos.
- Si no se recibe una señal después de un tiempo límite, se asume que el nodo ha fallado.
- Los nodos supervisan su tabla finger para detectar nodos inaccesibles y actualizarla si es necesario.

Recuperación de fallos:

Fallo de un nodo responsable de datos:

- Las réplicas de los datos almacenados en los sucesores se activan para mantener el acceso a los datos.
- El nodo más cercano al fallido asume temporalmente su responsabilidad.

Nivel de tolerancia a fallos esperado

El nivel de tolerancia depende de los siguientes factores:

Factores configurables:

1. Cantidad de réplicas ((k)):
    - Aumentar (k) mejora la tolerancia a fallos, pero consume más almacenamiento.
2. Frecuencia de avisos de actividad de los nodos:
    - Mensajes más frecuentes detectan fallos rápidamente, pero generan más tráfico.

Escenarios manejables:

- Nodos individuales:

    - Si un nodo falla, sus sucesores inmediatos pueden manejar sus responsabilidades temporalmente.
- Múltiples fallos concurrentes:
    - Si los fallos afectan a nodos consecutivos, el sistema puede degradarse, pero la probabilidad se reduce con un número mayor de réplicas.

Manejo de fallos parciales

Nodos caídos temporalmente:

1. Reintegración:
    - Cuando un nodo vuelve al sistema, consulta a sus sucesores y predecesores para recuperar datos perdidos.
    - Las tablas finger de otros nodos se actualizan dinámicamente para reflejar su reintegración.
2. Marcado de nodos inactivos:
    - Los nodos caídos temporalmente se marcan como inactivos en las tablas finger para evitar consultas innecesarias.

Incorporación de nodos nuevos:

1. Proceso de unión:
    - El nodo calcula su posición en el anillo y notifica a su sucesor inmediato.
    - El sucesor transfiere los datos que ahora son responsabilidad del nuevo nodo.
2. Redistribución de datos:
    - Los datos almacenados en nodos anteriores al nuevo nodo deben ser migrados a este.
    - Esto incluye tanto los datos originales como las réplicas.
3. Actualización de tablas finger:
    - Todos los nodos afectados actualizan sus tablas finger para incluir al nuevo nodo.

Casos de uso específicos

Fallo durante una operación:

- Subida de archivo:
    - Si el nodo responsable falla durante la subida, el cliente puede reenviar la solicitud al sucesor siguiente utilizando su tabla finger.
- Búsqueda de archivo:
    - Si un nodo intermedio en la ruta de búsqueda falla, el nodo anterior puede usar su tabla finger para reenviar la consulta.

- Descarga de archivo:
    - Si el nodo responsable del archivo está caído, el cliente puede acceder a una réplica en un sucesor.