SISTEMA DE ARCHIVOS DISTRIBUIDOS

POR:

Katherine Benjumea Ortiz

Brigith Lorena Giraldo Vargas

Valentina Ochoa Arboleda

PARA:

Edwin Nelson Montoya Acevedo

Docente

Tópicos especiales en telemática

Ingeniería de sistemas

EAFIT

Medellín - Colombia

2023

Contenido

Especificaciones:

1. Descripción arquitectura

Para el proyecto se tiene una arquitectura híbrida que combina elementos de cliente-servidor y peer-to-peer para implementar un sistema de archivos distribuidos, esta arquitectura combina la centralización del NameNode para llevar un registro de la ubicación de los archivos y la descentralización de los DataNodes para el almacenamiento físico de los archivos. El enfoque peer-to-peer se refleja en la distribución de los DataNodes, cada uno con su propio almacenamiento, mientras que el servidor NameNode actúa como un punto centralizado de coordinación**.**

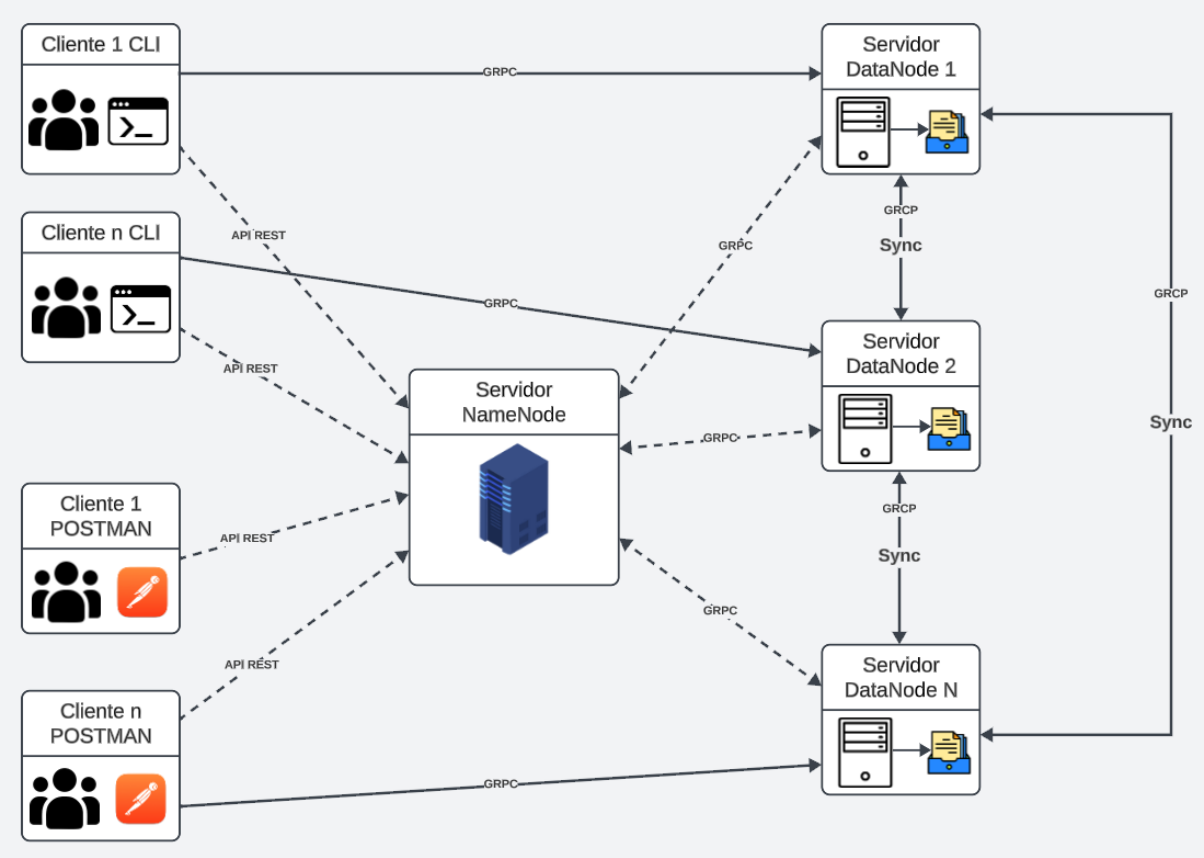

Imagen 1. Arquitectura DFS

Esto permite un equilibrio entre la escalabilidad y la capacidad de gestión centralizada en un sistema de archivos distribuidos. Esta combinación proporciona una serie de ventajas importantes:

- ***Gestión Centralizada de Metadatos***

En un sistema de archivos distribuidos, la gestión de metadatos (información sobre archivos y su ubicación) es fundamental. La arquitectura cliente-servidor permite que un servidor central, administre y mantenga los metadatos. Esto simplifica la gestión y la coherencia de los metadatos en todo el sistema.

- ***Escalabilidad y Rendimiento***

La arquitectura cliente-servidor del sistema se encuentra diseñada para lograr una escalabilidad horizontal efectiva. Esto mediante la capacidad de agregar servidores DataNode adicionales según las. Esta capacidad de escalabilidad permite aumentar la capacidad de almacenamiento y el rendimiento de manera eficiente. Cada servidor DataNode tiene la capacidad de almacenar y gestionar una porción específica de los archivos del sistema, esto permite distribuir la carga de trabajo de manera equitativa y eficiente entre los diferentes nodos del sistema.

- ***Tolerancia a Fallos***

En caso de fallos en un componente, como un DataNode o un cliente, el servidor central (por ejemplo, el NameNode) puede tomar medidas para mantener la disponibilidad y la redundancia de los datos. Puede gestionar la replicación de archivos y la reasignación de tareas en caso de fallos.

- ***Consistencia y Coherencia de Datos***

La arquitectura cliente-servidor permite mantener la consistencia y la coherencia de los datos, ya que el servidor central (por ejemplo, el NameNode) puede gestionar la asignación de bloques de datos y garantizar que los datos se actualicen de manera controlada y coherente.

- ***Gestión de Recursos***

La arquitectura cliente-servidor facilita la gestión de recursos, como el espacio de almacenamiento y la asignación de tareas. El servidor central puede supervisar y gestionar el uso de recursos en todo el sistema.

- ***Simplificación del Cliente***

Los clientes pueden ser más simples en una arquitectura cliente-servidor, ya que no necesitan gestionar metadatos complejos ni lidiar con decisiones de asignación de bloques. Simplemente envían solicitudes al servidor y se encargan de la transferencia de datos.

- ***Flexibilidad de Implementación***

La arquitectura cliente-servidor permite implementar diferentes componentes en diferentes lenguajes de programación o sistemas operativos según las necesidades y la infraestructura existente.

Por lo tanto, la combinación de la arquitectura cliente-servidor en la capa de control y la comunicación peer-to-peer (P2P) en la transferencia de archivos aporta una serie de ventajas significativas. Mientras que la arquitectura cliente-servidor brinda una gestión centralizada de metadatos, escalabilidad y tolerancia a fallos, la comunicación P2P descentraliza la transferencia de archivos, mejora la eficiencia del ancho de banda, reduce la dependencia del servidor central y aumenta la resiliencia de la red. En conjunto, esta arquitectura híbrida permite una gestión eficaz de metadatos, seguridad, escalabilidad y tolerancia a fallos, al mismo tiempo que aprovecha la eficiencia y la distribución de recursos proporcionada por la comunicación P2P para la transferencia de archivos, lo que resulta fundamental en entornos distribuidos donde se requieren confiabilidad y eficiencia.

2. Componentes

- **Cliente:**

El cliente es la interfaz a través de la cual los usuarios interactúan con el sistema de archivos distribuidos. El cliente puede ser una aplicación de línea de comandos o API externa (POSTMAN) que permite a los usuarios realizar diversas operaciones en archivos, como escribir, leer, actualizar y listar archivos.

- **Servidor NameNode**:

El servidor NameNode actúa como un punto centralizado de coordinación en la arquitectura. Su función principal es llevar un registro de la ubicación de los archivos en el sistema y la ubicación de los DataNodes que almacenan esos archivos. Los DataNodes se registran en el NameNode para que este último conozca su existencia y disponibilidad.

- **Servidores DataNode**:

Los servidores DataNode son donde se almacenan los archivos en el sistema. Cada DataNode es responsable de gestionar y almacenar los archivos que se le asignan. Estos servidores se registran en el NameNode para que este último pueda realizar un seguimiento de su disponibilidad y la ubicación de los archivos.

3. Comunicaciones

- **Cliente a NameNode:**

Para la comunicación entre cliente y nameNode se decide implementar API REST por su simplicidad, escalabilidad, independencia de plataforma, seguridad, facilidad de documentación y pruebas, interoperabilidad y facilidad de mantenimiento. Esta decisión complementa la arquitectura híbrida del sistema, que ya integra elementos cliente-servidor y comunicación peer-to-peer en la transferencia de archivos, proporcionando un enfoque eficiente y estándar para las interacciones cliente-NameNode. Esto facilita la implementación, escalabilidad y evolución del sistema, promoviendo una comunicación confiable en un entorno distribuido donde la eficiencia y la compatibilidad son fundamentales.

- **Cliente a DataNode:**

Se decide implementar el protocolo GRPC para la comunicación entre el Cliente y el DataNode lo que proporciona una comunicación eficiente, segura y tipada en una arquitectura híbrida de sistemas de archivos distribuidos. Este enfoque garantiza que las operaciones del Cliente se ejecuten de manera confiable en el DataNode correspondiente y que cualquier resultado o error se comunique de manera efectiva al Cliente, lo que contribuye a la confiabilidad y la eficiencia del sistema en su conjunto.

- **NameNode A NameNode:**

Para la comunicación entre los NameNodes en la arquitectura actual, se va a utilizar gRPC, debido a que es una tecnología de comunicación es eficiente y versátil que se adapta bien a escenarios de comunicación entre servidores y que puede gestionar fácilmente la comunicación bidireccional, lo cual es necesario en la arquitectura que se va a manejar, donde se menciona que la comunicación entre los NameNodes se da en doble sentido.

Además, con gRPC, es posible definir las interfaces de servicio y los métodos que los NameNodes pueden invocar entre sí, lo que facilita la sincronización y la gestión de metadatos.

- **DataNode A DataNode:**

En la comunicación entre DataNode a DataNode en la arquitectura híbrida se utiliza el protocolo gRPC, el cual se debe de implementar definiendo servicios y métodos gRPC específicos para la transferencia eficiente de datos entre nodos. Se debe abordar la gestión de errores y la confiabilidad en la implementación, considerando la detección de fallos y la replicación de datos según sea necesario para garantizar la integridad de la comunicación P2P entre DataNodes en el sistema de archivos distribuidos.

- **NameNode a DataNode:**

En la comunicación entre el NameNode y los DataNodes se utiliza el grpc y este se justifica por su eficiencia, generación de código automático, interoperabilidad, seguridad y documentación clara. Además, las definiciones de servicio en Protocol Buffers garantizan una comunicación sólida y tipada, y su compatibilidad con múltiples lenguajes facilita la interoperabilidad. En conjunto, estas ventajas hacen que gRPC sea la elección adecuada para una comunicación eficiente y segura en la arquitectura actual del sistema de archivos distribuidos.

- **Archivo proto:**

Para la implementación del protocolo gRPC es esencial definir la estructura base de los archivos proto que especifican los mensajes y servicios que se utilizarán en la comunicación entre los nodos. Estos archivos proto actúan como el contrato que los nodos deben seguir para intercambiar información de manera efectiva. Además, gRPC permite la implementación de transmisiones bidireccionales mediante un método rpc específico en el servicio gRPC, lo que posibilita que ambos extremos envíen y reciban datos simultáneamente, lo que es fundamental para la interacción en tiempo real y la sincronización entre los nodos del sistema de archivos distribuidos.

Se han definido dos servicios en el archivo proto del proyecto:

-NameNode: Este servicio define cinco operaciones remotas (RPC) que pueden ser llamadas desde otros componentes del sistema:

- RegisterDataNode: Permite a un nodo de datos registrarse en el sistema.
- GetFileLocation: Permite obtener la ubicación de un archivo.
- WriteFile: Permite escribir datos de archivo en el sistema (aunque en la declaración del servicio, la respuesta es de tipo GetFileLocationResponse, lo que podría ser un error).
- ReadFile: Permite leer datos de un archivo.
- InformFileLocation: Permite informar la ubicación de un archivo a un nodo.

-DataNode: Este servicio define dos operaciones remotas:

- StoreFile: Permite a un nodo de datos almacenar un archivo en el sistema.
- ReadFile: Permite a un nodo de datos leer un archivo del sistema.

4. Replicación

En este proyecto, la replicación de los DataNodes se implementa a través de gRPC con el objetivo principal de garantizar la disponibilidad y la tolerancia a fallos en el sistema de almacenamiento distribuido. La replicación se lleva a cabo entre un DataNode líder y un DataNode seguidor, asegurando que un archivo se almacene de manera redundante en ambos nodos. Esta estrategia tiene un propósito crítico: en caso de que el DataNode líder experimente una falla, los archivos que contiene no se perderán, ya que estarán respaldados en el DataNode seguidor. Esta arquitectura mejora significativamente la confiabilidad del sistema y permite una gestión más eficaz de los datos distribuidos.