

Informe - Red Social

Eduardo Brito Labrada (C411)

Ernesto Abreu Peraza (C412)

1. Arquitectura

El proyecto consiste en implementar una red social distribuida para la comunicación siguiendo la idea en la cual se base Twitter. El sistema debe brindar una infraestructura para el manejo de usuarios, de manera que puedan publicar sus mensajes, volver a publicar mensajes que consideren interesantes de otros usuarios y escoger quienes van a ser sus amigos. Además, como parte del manejo de usuarios, se debe proporcionar un mecanismo para la autenticación distribuida de manera que se logre un modelo que no sea centralizado.

Para la arquitectura, nos basaremos en una red de servidores y clientes que se comunican a través de gRPC. Los servidores están organizados en un anillo de Chord para la distribución y almacenamiento de datos. A continuación se especifica en más detalle el funcionamiento de cada parte del sistema:

- Cliente: se encarga de localizar un servidor activo en la red para enviar solicitudes. Al servidor localizado le solicita la publicación de posts, seguimiento de usuarios o autenticación.
- Servidor: se encarga de recibir y procesar solicitudes que le llegan de los clientes. Además, es quien ofrece los servicios requeridos por el sistema como publicación de posts, autenticación de los usuarios y manejo de relaciones entre los usuarios.
- Repositorio de Datos: es el intermediario que se encarga de gestionar el almacenamiento y la recuperación de los datos. Se comunica con los nodos del anillo de Chord para realizar el almacenamiento distribuido.
- Nodos en el anillo de Chord: cada nodo almacena datos según el hash calculado para la llave del post o usuario. Implementa las funcionalidades específicas de Chord para distribuir y localizar datos.

Los servicios estarán distribuidos en dos redes de Docker:

- Red de clientes: alojará múltiples instancias de clientes que interactúan directamente con los usuarios, estas instancias no comparten información entre sí y dependen completamente de la red de servidores para procesar cada uno de los servicios del sistema.
- Red de servidores: contiene los contenedores que ejecutan los servicios del servidor, incluyendo el repositorio de datos y los nodos en el anillo de Chord. Los servidores se comunican entre sí para mantener la consistencia y la disponibilidad de los datos en todo momento.

Las redes de Docker separadas para clientes y servidores garantiza una segregación adecuada del tráfico y mejoran la seguridad general del sistema.

2. Procesos

Los principales procesos necesarios para este sistema son:

1. Los clientes, que se encargan de manejar las solicitudes de los usuarios.
2. Los servidores, que se encargan de ejecutar servicios específicos que requiere el sistema.
3. Un repositorio de datos que actúa como intermediario para gestionar almacenamiento y recuperación de datos.
4. Los nodos del anillo de Chord que se encargan de distribuir y almacenar los datos hashados.

Cada uno de estos servicios opera de manera asíncrona. Se utilizarán hilos para manejar las solicitudes concurrentes, lo que permite escalabilidad y tolerancia a fallos, empleando redes de Docker separadas para aislar la comunicación entre clientes y servidores. Esto permite un desempeño eficiente en entornos distribuidos al garantizar una interacción fluida y segura entre las distintas componentes del sistema.

3. Comunicación

Para la comunicación entre el cliente y el servidor, se utilizará gRPC como protocolo principal, lo cual brinda una serialización eficiente y un modelo de comunicación robusto. Además, brinda una interfaz clara y tipada para las interacciones lo cual facilita el manejo de las operaciones síncronas como autenticación de usuarios y publicación de posts.

La comunicación entre servidores dentro del anillo de Chord será implementada mediante sockets lo que permite una gestión eficiente de la distribución de información entre los nodos y proporciona mayor control en la transferencia de datos. Con los sockets se pueden mantener conexiones persistentes entre los nodos del anillo de Chord, lo que hace más fácil la propagación de actualizaciones y la sincronización del estado distribuido.

La comunicación entre los procesos se gestionará a través de la red de Docker, que proporciona el aislamiento necesario además de una conectividad fluida entre los contenedores.

Este modelo de comunicación permite sacar provecho de las fortalezas de cada protocolo: gRPC para interfaces estructuradas cliente-servidor y sockets para la comunicación eficiente entre los nodos del anillo de Chord.

4. Coordinación

La coordinación entre los servicios se manejará mediante mecanismos de sincronización y consenso para garantizar acceso exclusivo a recursos compartidos y evitar condiciones de carrera. Se utilizarán algoritmos distribuidos como el de exclusión mutua distribuida, que asegura que un solo nodo pueda modificar un recurso en el momento dado, junto a un protocolo de toma de decisiones distribuida para coordinar acciones críticas como la actualización de datos o manejo de fallos, permitiendo que los nodos siempre se mantengan consistentes en el sistema, además siendo de forma eficiente y confiable.

5. Nombrado y Localización

La localización de recursos se hará mediante el anillo de Chord, que asigna a cada recurso con un nodo específico teniendo en cuenta un hash único. Los datos y servicios se identifican por las llaves hash obtenidas de identificadores únicos lo que garantiza una distribución uniforme. Para ubicar y acceder a los datos o servicios, el cliente realiza una solicitud que desde el anillo de Chord llega al nodo correspondiente. Este enfoque asegura

que la localización sea eficiente y escalable, después de cambios dinámicos en la red.

6. Consistencia

El sistema utilizará un modelo de consistencia eventual, garantizando que todas las réplicas lleguen hacia el mismo estado después de un tiempo, incluso si las actualizaciones se hacen de forma concurrente. La distribución de los datos se basa en el anillo de Chord, donde los datos son asignados a nodos según el hash de su llave. Este enfoque garantiza un balance adecuado de la carga y minimiza el riesgo de sobrecarga en nodos individuales, lo que hace que se tenga una replicación eficiente.

Para garantizar alta disponibilidad, cada dato será replicado en $k = 3$ nodos consecutivos del anillo, el valor de k es variable pero como en este caso solo se necesita tolerancia a fallos 2, con $k = 3$ se consigue garantizar esa confiabilidad del sistema. Esta replicación protege contra fallos de nodos individuales y asegura que aún teniendo nodos fuera de servicio el sistema pueda seguir respondiendo las solicitudes del cliente.

Al realizar una actualización, el nodo responsable de la llave propaga los cambios a sus réplicas. Un protocolo de confirmación asegurará que las actualizaciones sean reconocidas por varias réplicas antes de ser consideradas como completadas. Este enfoque fortalece la confiabilidad de los datos y garantiza que las actualizaciones sean consistentes aún con posibles fallas.

7. Tolerancia a fallos

En caso de que un nodo del anillo de Chord deje de estar disponible, su ausencia no afecta el funcionamiento del sistema. Los datos almacenados en dicho nodo estarán replicados en $k = 3$ de sus vecinos para garantizar su recuperación y disponibilidad. Esto asegura que las solicitudes de los clientes puedan ser atendidas incluso ante fallos parciales de nivel $k - 1$.

El sistema admite incorporación de nuevos nodos en tiempo de ejecución. Para cada nodo se calcula su posición en el anillo utilizando el hash correspondiente y redistribuye las llaves relevantes de forma automática, minimizando la pérdida de datos.

Cuando un cliente intenta interactuar con un nodo inactivo, el sistema redirige automáticamente la solicitud a otro nodo que esté activo mediante la estructura del anillo de

Chord. Además, se implementarán mecanismos de reintentos y reconexión para mantener la confiabilidad.