ARQUITECTURA

El diseño de este sistema parte de la necesidad de crear una infraestructura que permita la colaboración entre usuarios distribuidos, asegurando la integridad de los datos y una respuesta eficiente a las solicitudes. La arquitectura peer-to-peer estructurada proporciona una solución ideal al permitir la distribución de responsabilidades entre nodos de forma equitativa, reduciendo puntos únicos de fallo y mejorando la escalabilidad. Cada nodo en este sistema representa a un usuario y actúa como cliente y servidor al mismo tiempo, procesando solicitudes locales y remotas, y almacenando una parte de la información global.

Chord se utiliza como protocolo de enrutamiento y localización de datos, organizando los nodos en un anillo lógico donde cada uno es responsable de un rango específico de claves. Esto permite una búsqueda eficiente de eventos o usuarios con una complejidad logarítmica en el número de nodos. Adicionalmente, el sistema implementa mecanismos para manejar la unión y salida de nodos, asegurando la consistencia del anillo y la redistribución adecuada de datos.

Al ser cada nodo un cliente y servidor, los usuarios pueden gestionar sus propias agendas y también colaborar con otros al procesar solicitudes distribuidas. Los datos se fragmentan y distribuyen entre los nodos del anillo, garantizando redundancia y tolerancia a fallos mediante la replicación en sucesores cercanos.

La comunicación entre nodos se realiza mediante mensajes estructurados que permiten consultar, agregar o modificar eventos. Cada nodo valida localmente las operaciones antes de propagarlas al resto del sistema, asegurando que las agendas individuales y compartidas se mantengan consistentes. Este enfoque también permite que el sistema escale de manera natural a medida que se agregan más usuarios.

El sistema define varios roles clave para gestionar las interacciones y garantizar su funcionamiento eficiente. Cada nodo tiene capacidades equivalentes en cuanto a almacenamiento y procesamiento de datos, pero los usuarios que representan pueden asumir diferentes niveles jerárquicos. Los usuarios finales interactúan con el sistema para crear, modificar o consultar eventos en sus agendas personales o colectivas. Los moderadores tienen la responsabilidad de validar eventos propuestos por usuarios de menor jerarquía y resolver conflictos en agendas compartidas.

Para facilitar el despliegue y la gestión del sistema, se utilizan dos redes Docker: una interna y otra externa. La red interna conecta los contenedores que implementan los nodos del anillo Chord, permitiendo una comunicación eficiente y segura entre ellos. Esta red se utiliza para operaciones sensibles, como la redistribución de datos y la sincronización de estado entre nodos.

La red externa expone los servicios que requieren interacción directa con los usuarios o sistemas externos. Estos servicios incluyen la autenticación de usuarios, la interfaz para la gestión de agendas y la notificación de eventos. Cada nodo actúa como un punto de entrada al sistema, permitiendo que las solicitudes de usuarios se procesen localmente o se redirijan a otros nodos según sea necesario. Los contenedores en esta red están configurados para comunicarse con los nodos internos a través de una API segura, garantizando que las operaciones de usuario se traduzcan correctamente en cambios dentro del anillo.

PROCESOS

Los procesos se diseñaron para abarcar todas las funcionalidades principales del sistema y garantizar la sincronización, privacidad y consistencia de los datos. Los principales son:

- Gestión de Contactos: Permite agregar, eliminar o buscar contactos en el sistema. Sincroniza los cambios de la lista de contactos entre nodos participantes. Verifica la privacidad y autorización antes de compartir información de contacto.

- Gestión de Grupos: Soporta la creación, modificación y eliminación de grupos. Implementa jerarquías dentro de los grupos (administradores, miembros regulares). Sincroniza la membresía y configuración del grupo entre nodos. Permite visualizar agendas grupales y gestionar eventos grupales.

- Gestión de Eventos (Creación, Modificación y Confirmación): Procesa la creación de eventos personales y grupales. Verifica conflictos de horario. Asegura que las reglas de privacidad y jerarquías se respeten. Permite confirmar eventos pendientes de aceptación (para eventos grupales y personales con participantes).

- Sincronización de Datos Entre Nodos: Propaga cambios de eventos entre los nodos participantes. Usa una tabla hash distribuida (DHT) basada en Chord para localizar y sincronizar datos.

- Visualización de Agendas: Genera vistas combinadas de agendas personales y grupales en forma de línea de tiempo. Filtra los detalles de eventos según las reglas de privacidad (muestra si un usuario está ocupado si el solicitante no es participante; proporciona detalles completos del evento si el solicitante es participante.)

- Notificaciones: Envía invitaciones a eventos grupales. Notifica actualizaciones o cancelaciones de eventos.

- Autenticación y Seguridad: Valida usuarios y nodos que interactúan en la red. Protege datos sensibles mediante cifrado en tránsito y almacenamiento local.

- Mantenimiento de la Red Distribuida: Gestiona la conexión y desconexión de nodos. Reasigna la responsabilidad de datos en la DHT cuando un nodo entra o sale de la red.

La organización de los procesos en el sistema de agenda distribuida se estructura de manera que cada nodo combina funcionalidades de cliente y servidor, operando como una instancia autónoma. Dentro de cada nodo, los procesos se agrupan en módulos funcionales: gestión de eventos, sincronización de datos, administración de grupos y contactos, notificaciones y visualización de agendas. Estos procesos interactúan de forma local para validar datos, manejar actualizaciones y respetar las reglas de privacidad, mientras que a nivel distribuido, los nodos intercambian información mediante una arquitectura basada en DHT para garantizar la consistencia de los datos.

Este enfoque modular permite que cada instancia funcione de manera independiente, pero colabore con otras para sincronizar eventos, resolver conflictos y respetar jerarquías, logrando así un sistema escalable, tolerante a fallos y alineado con los principios de descentralización.

El sistema de agenda distribuida emplea un enfoque híbrido para garantizar un desempeño eficiente y escalable. Para las operaciones locales, como validación de datos y gestión de la base de datos, se utiliza un modelo de hilos ligeros que permite procesar múltiples solicitudes de manera concurrente sin bloquear la ejecución principal.

En las interacciones entre nodos, se adopta un patrón asincrónico basado en sockets TCP/IP, permitiendo una comunicación bidireccional y confiable en tiempo real. Esto optimiza la latencia y asegura una rápida sincronización de datos entre los participantes. Además, para operaciones intensivas o independientes, como la resolución de conflictos y la propagación de eventos en la red distribuida, se implementan procesos separados que operan en paralelo. Este enfoque combina lo mejor de los patrones asincrónicos, hilos y procesos, equilibrando el uso de recursos y garantizando una experiencia fluida en un entorno descentralizado.

COMUNICACIÓN

En una arquitectura peer-to-peer basada en Chord, la distinción entre cliente y servidor se difumina, ya que cada nodo desempeña ambos roles. La comunicación cliente-servidor se manifiesta cuando un nodo realiza una solicitud, como buscar un evento en la agenda. En este caso, el uso de HTTPS garantiza la seguridad y confidencialidad de las transacciones.

Por otro lado, la comunicación servidor-servidor es fundamental para mantener la consistencia y la distribución de datos en la red Chord. ZMQ destaca por su simplicidad y velocidad, ideal para implementar patrones de comunicación como pub-sub o push-pull en la sincronización de datos entre nodos por lo que podría ser preferible debido a su eficiencia en la transferencia de mensajes en tiempo real entre nodos.

La comunicación entre procesos dentro de un nodo también es un componente esencial del diseño. En el proyecto de la agenda, donde un nodo puede tener hilos o procesos dedicados a validar datos, actualizar la base de datos y enviar notificaciones, la elección del mecanismo de comunicación es crucial. Los semaforos representan una solución sencilla y eficaz para coordinar flujos de trabajo secuenciales entre procesos. Permiten gestionar el acceso a recursos compartidos entre procesos locales o remotos, asegurando que solo un proceso pueda acceder a un recurso en un momento dado, evitando condiciones de carrera y garantizando que los datos se transmitan de manera estructurada y eficiente.

COORDINACIÓN

En el sistema de agenda distribuida, la sincronización de acciones es esencial para garantizar que los datos estén consistentes entre todos los nodos participantes. Por ejemplo, cuando un usuario crea o actualiza un evento compartido, el nodo de origen propaga los cambios a través de la red distribuida utilizando un protocolo basado en marcas de tiempo (last_updated). Este mecanismo asegura que cada nodo aplique las actualizaciones en el orden correcto, evitando sobrescrituras o inconsistencias.

Para garantizar que la información esté sincronizada en tiempo real, se utilizan sockets TCP/IP, lo que permite establecer conexiones confiables y persistentes entre los nodos de la red. Estos sockets aseguran que las actualizaciones bidireccionales puedan enviarse y recibirse de manera eficiente, manteniendo a todos los participantes involucrados en eventos compartidos trabajando con la misma información actualizada. Si un nodo se desconecta temporalmente, recuperará las actualizaciones pendientes cuando vuelva a estar en línea, gracias a la arquitectura basada en DHT, como Chord.

El acceso exclusivo a recursos en un sistema distribuido requiere manejar posibles condiciones de carrera que podrían surgir cuando múltiples nodos intentan modificar los mismos datos simultáneamente. Para abordar este desafío, el sistema implementa un esquema de control de concurrencia basado en relojes lógicos y un mecanismo de bloqueo distribuido. Los nodos utilizan marcas de tiempo para identificar el orden de las operaciones, resolviendo conflictos al priorizar la operación más reciente. De forma adicional, las actualizaciones críticas, como la confirmación de eventos o cambios en la configuración de grupos, se protegen mediante un protocolo de consenso implícito en el que los nodos participantes verifican las modificaciones antes de aceptarlas. Este enfoque asegura integridad en los datos y elimina conflictos derivados del acceso simultáneo.

La toma de decisiones distribuidas es fundamental en un sistema donde no existe una entidad central para coordinar las acciones. Por ejemplo, al organizar un evento grupal, el nodo del organizador envía una solicitud a todos los nodos participantes. Cada nodo evalúa la solicitud en función de su disponibilidad y responde con una aceptación o rechazo. El consenso se logra mediante un protocolo de votación, donde el evento se confirma cuando todos los participantes aceptan, o si las reglas predefinidas lo permiten (por ejemplo, en grupos jerárquicos donde los eventos creados por un administrador se confirman automáticamente). Este mecanismo descentralizado, apoyado por la arquitectura DHT, permite que el sistema mantenga una alta disponibilidad y sea tolerante a fallos, incluso si algunos nodos se desconectan o fallan temporalmente.

NOMBRADO Y LOCALIZACIÓN

La identificación de los datos y servicios en un sistema distribuido es el primer paso para garantizar que puedan ser accedidos y utilizados de manera eficiente. En una red como la de la agenda distribuida, cada recurso, ya sea un evento, un usuario o un nodo, debe contar con un identificador único. Chord implementa este principio utilizando funciones hash consistentes para asignar identificadores tanto a los nodos como a los recursos. Por ejemplo, un evento en la agenda podría identificarse mediante un hash derivado de su nombre, fecha y creador, lo que asegura unicidad y permite organizar los datos de forma distribuida en el anillo lógico de la red.

La ubicación de los datos y servicios es el siguiente desafío. En un sistema distribuido, los datos no están centralizados, sino que se distribuyen entre los nodos de la red. En la arquitectura Chord, la ubicación de un recurso está determinada por su identificador hash. Cada nodo en el anillo lógico es responsable de un rango específico de identificadores, lo que asegura una distribución equilibrada de los recursos. Por ejemplo, si un usuario crea un nuevo evento en la agenda, el identificador hash del evento determinará en qué nodo se almacenará. Este nodo será responsable de responder a las solicitudes relacionadas con ese evento.

La localización de los datos y servicios se refiere al proceso de encontrar el nodo que almacena un recurso específico y establecer una conexión para acceder a él. En Chord, este proceso se realiza mediante un algoritmo de enrutamiento eficiente que utiliza tablas de finger (finger tables). Por ejemplo, si un usuario A desea consultar un evento creado por el usuario B, el sistema primero debe localizar el nodo que almacena el evento mediante el protocolo Chord. Una vez localizado, el usuario A puede enviar una solicitud HTTPS para recuperar los detalles del evento.

Además, el sistema debe ser capaz de manejar dinámicamente la adición y eliminación de nodos sin comprometer la consistencia ni la localización de los recursos. Chord aborda este desafío actualizando las tablas de finger y reasignando los recursos afectados cuando un nodo se une o abandona la red.

CONSISTENCIA Y REPLICACIÓN

En el sistema de agenda distribuida, la consistencia y replicación son aspectos clave para garantizar que los datos estén disponibles y sean confiables en un entorno distribuido. Los datos se distribuyen utilizando una DHT basada en Chord, donde cada nodo almacena una porción específica de la información y mantiene réplicas en sus nodos vecinos inmediatos. Esto asegura que los datos permanezcan accesibles incluso si uno o varios nodos fallan o se desconectan. Para mejorar la disponibilidad y tolerancia a fallos, cada dato se replica en un número configurable de nodos (k réplicas), de modo que siempre existan varias copias disponibles en la red. Para nuestro proyecto de agenda distribuida, un valor de k=3 parece adecuado porque proporciona un buen balance entre disponibilidad y uso de recursos, tolera fallos de hasta 2 nodos sin perder datos.

El sistema implementa un modelo de consistencia eventual, lo que significa que las actualizaciones realizadas en un nodo se propagan de forma asíncrona a todas las réplicas. Para garantizar que las réplicas estén actualizadas y evitar inconsistencias, se utiliza un mecanismo basado en marcas de tiempo (last_updated) para identificar y aplicar siempre la versión más reciente del dato como mencionamos anteriormente. Además, en caso de conflictos, las marcas de tiempo permiten resolverlos automáticamente, asegurando que el sistema funcione de manera coherente.

La confiabilidad de las réplicas se refuerza a través de procesos periódicos de sincronización entre nodos, que verifican la integridad de los datos y corrigen posibles desajustes. Este enfoque equilibra la necesidad de consistencia con el rendimiento del sistema, permitiendo que los usuarios accedan a datos precisos sin experimentar retrasos significativos. Con esta estrategia, el sistema asegura que tanto la distribución como la replicación de los datos sean robustas y escalables, adaptándose a las características dinámicas de una red distribuida.

TOLERANCIA A FALLAS

La respuesta a errores en un sistema distribuido implica diseñar mecanismos que detecten, manejen y, en la medida de lo posible, mitiguen los efectos de las fallas. En la agenda distribuida, esto se logra mediante varias estrategias. Por ejemplo, Chord implementa redundancia en el almacenamiento de datos al replicar recursos en múltiples nodos sucesores en el anillo lógico. Si un nodo falla, los datos aún pueden recuperarse de sus copias en los nodos vecinos. Además, el protocolo Chord incluye mecanismos de detección de fallos mediante el monitoreo de los enlaces entre nodos. Si un nodo no responde, su responsabilidad se transfiere automáticamente a otro nodo, minimizando el impacto en la red.

Los fallos parciales son una característica inherente de los sistemas distribuidos. En el contexto de nuestro proyecto los nodos pueden caer temporalmente debido a problemas de conectividad, mantenimiento o errores de hardware. Para manejar estos casos, Chord utiliza un protocolo de estabilización que actualiza las tablas de finger y las responsabilidades de los nodos cuando se detecta un cambio en la topología de la red. Por ejemplo, si un nodo se desconecta, sus recursos se redistribuyen automáticamente entre los nodos restantes, asegurando que sigan siendo accesibles. Cuando un nodo se reincorpora, puede recuperar su rango de responsabilidad y reintegrarse al anillo sin afectar el funcionamiento general del sistema.

La incorporación de nuevos nodos también es un aspecto crítico en la tolerancia a fallas. En nuestro calendario, los nuevos nodos pueden unirse a la red de manera dinámica sin necesidad de detener el sistema. Chord permite que un nodo recién llegado se conecte al anillo lógico y obtenga su rango de identificadores hash de manera eficiente. Durante este proceso, el nodo también replica los datos necesarios para garantizar la redundancia y la disponibilidad. Este enfoque asegura que la red pueda escalar de manera eficiente y adaptarse a cambios en la carga o en el número de usuarios.