Aplicación para la gestión de clientes en un banco

Vamos a tener 3 réplicas (app distribuida con varias réplicas) cada una con sus datos y tenemos clientes que se conectan con sus réplicas (no balanceo, el cliente conoce a que replica conectarse)
**Es una replicación activa lo que hay que hacer -> Si tengo 3 réplicas, cuando un cliente manda una operación a una de ellas, esa tiene que reenviarla a todas las demás réplicas. Comunicación fiable y con orden total

Objetivos

- Gestiona la información de los clientes de un banco
- La aplicación debe ser tolerante a fallos y coherente

El interfaz le da igual, le importa lo que hay debajo
Los clientes pueden ejecutar operaciones sobre las réplicas y se deben replicar en todos para garantizar la coherencia
En este caso deja de ser configuración standalone y pasa a ser quorum
En todas las répllicas debe haber la misma información aunque el cliente solo hable con una
Es muy dificil mirar el codigo sin saber que voy a hacer, si me pierdo en el código volver para atrás

Introducción

El objetivo de esta tarea es desarrollar una aplicación que gestione la información de los clientes de un banco. La aplicación debe ser distribuida, para asegurar que las propiedades de consistencia, disponibilidad y tolerancia a fallos. Las funcionalidades o servicios son sencillos, para concentrarse en los aspectos distribuidos del sistema.

La gestión de los clientes se basa en funcionalidades sencillas:

- La información que debe conservarse para cada cliente incluye su número de cuenta, nombre y saldo.
- Gestionar una base de datos, encapsulando la información de los clientes.
- Las operaciones o servicios que deben proporcionarse para gestionar la información de un cliente son: crear un cliente, obtener su información, actualizar su saldo (depositar o retirar), y eliminar un cliente.

La siguiente figura describe la arquitectura general del sistema previsto. Los clientes interactúan con un servidor, que mantiene la información en su disco local. Un conjunto ZooKeeper proporciona los medios básicos para configurar una membresía servicio, basado en un detector de fallos, y permitir que los servidores garanticen la consistencia de las operaciones solicitadas por los clientes.

La base de datos va a estar en memoria, no es necesario mongodb ni nada

Requisitos

La aplicación deberá gestionar la información de los clientes del banco. El registro de cada cliente incluye su número de cuenta, su nombre y su depósito.

1. La aplicación proporcionará los siguientes servicios para la gestión de los clientes: **crear,leer, actualizar y eliminar.**
2. Inicialmente, la aplicación proporcionada es una interfaz simple, basada en texto. Los clientes podrían interactuar con esta API. El objetivo es concentrar el esfuerzo en los aspectos distribuidos.
3. La aplicación se basará en un **conjunto de servidores replicados** para proporcionar las propiedades distribuidas relevantes: consistencia, tolerancia a fallos y disponibilidad.
4. **Consistencia:** el estado de la información de los clientes será coherente en todos los servidores. En este caso, el estado debe mantenerse igual en cualquier momento. Todos tienen los mismos datos
5. **Tolerancia a fallos:** La aplicación debe tolerar (un número de) fallos. Se debe proporcionar la misma funcionalidad a pesar del límite de fallos. Si uno falla se puede replicar
6. **Disponibilidad:** Los gestores de los datos podrán invocar a cualquiera de los servidores replicados. Es aceptable un cierto retraso en el acceso a los servidores debido a las operaciones para garantizar la coherencia o la gestión de fallos. Tenemos muchos nodos y muchos clientes pueden acceder
7. La aplicación deberá **mantener el número de servidores correctos**. En caso de que un servidor fallara, se iniciará un nuevo servidor y se transferirá el estado del sistema al nuevo servidor, para mantener la consistencia dentro del sistema.
8. (Opcional) Utilizar una **base de datos** (MongoDB, SQL, etc.) para gestionar los datos.
9. (Opcional) Utilizar una **página web** para el acceso de los clientes a las operaciones disponibles.
10. (Opcional) Utilizar un **balanceador de carga** para permitir a los clientes acceder a los servidores con transparencia.

Parte 1: garantizar la coherencia de la información de los clientes.

Centrarse solo en la coherencia -> Replicación activa

La aplicación proporciona operaciones o servicios para que los clientes puedan acceder y actualicen su información. Cada cliente interactúa con un servidor para solicitar servicios.

Es obligatorio garantizar que la base de datos local de cada cliente sea coherente, es decir, que la información sea la misma en todos los servidores.

Parte 2: garantizar la tolerancia a fallos

Centrarse solo en la tolerancia a fallos

La aplicación tiene que garantizar que la aplicación será correcta, a pesar de hasta f fallos de proceso. Los clientes no pueden notar la diferencia a pesar de los fallos.

La aplicación distribuida estará compuesta por un número de servidores, que define su quórum. Si no se proporciona ese valor, las operaciones no podrán ejecutarse. Para

simplificar el desarrollo, se considera que no puede haber más servidores en ejecución que el quórum.

Después de que un servidor falle tras conseguir el quórum, el servidor que lo sustituya deberá obtener la base de datos correcta anterior.

Es opcional:

- Cuando se detecta el fallo de un servidor, el reemplazo debe iniciarse automáticamente.
  Hay que matar a una de las réplicas y arrancar una nueva para que siga funcionando
- La base de datos del servidor sustituto puede almacenarse en el mismo ordenador que el líder.

Comentarios sobre la asignación

- La capacidad máxima de memoria de un znode en un conjunto ZooKeeper es de 1Mb. Esto implica que la base de datos o los archivos para gestionar los datos de los clientes no se pueden almacenar en un znodo. ZooKeeper se utilizará para garantizar la coherencia y la coordinación entre los servidores.
- La coherencia de los datos de los servidores se basará exclusivamente en ZooKeeper. Un servidor enviará los datos a un nuevo servidor después, pero previamente las funciones basadas en ZooKeeper podrán acordar los servidores.

-Una operación se puede ejecutar cuando hayan terminado las anteriores
-Hay que difundir todos los datos a todos

Ejecución del programa

1. Descargar 2022_bank.tar y bank_library_2022.jar
2. Importar el fichero 2022_bank.tar en Eclipse
3. Incluir la biblioteca bank_library_2022.jar en el proyecto 2022_bank_assign
4. Ejecutar el programa MainBank.java
5. Por defecto, se usa la versión Java 13. Si el Eclipse no lo tiene, se puede seleccionar la versión adecuada. Desde el proyector, seleccionar:
Build Path -> Configure build path -> Libraries. Seleccionar la versión que se usa a la que se dispone.