PATRÓN DE DISEÑO CIRCUIT BREAKER

Arquitectura del Software

Gaspar Pisa Eyaralar - UO250825
Ignacio González Sánchez - UO260020

¿Qué es “Circuit breaker”?

Circuit breaker es un patrón de diseño centrado principalmente en evitar fallos en cascada provocados por un exceso de peticiones realizadas a un proveedor que no responde.

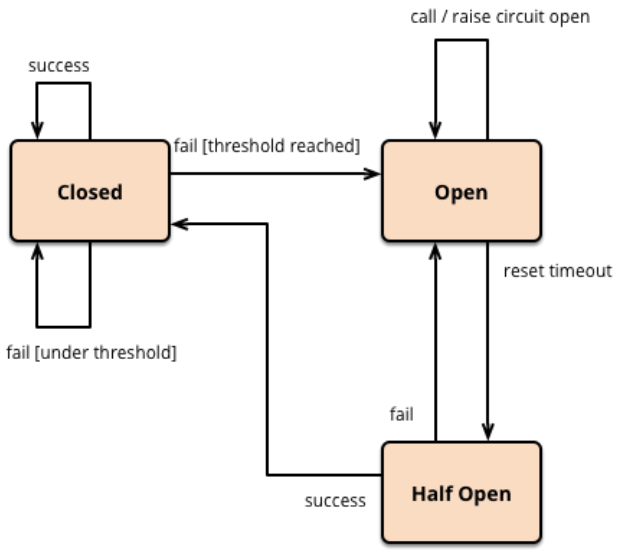
Este patrón en sistemas donde se realicen llamadas remotas, las cuales pueden fallar haciendo que se realice de nuevo dicha petición, si este proceso se da en muchas peticiones puede que nuestro sistema encargado de tratarlas se sobrecargue pudiendo provocar fallos en cascada en otros sistemas relacionados, para evitar este problema utilizamos el patrón “Circuit breaker”.

¿Cómo funciona el patrón “Circuit breaker”?

La idea del funcionamiento del patrón es recubrir la petición en un objeto “circuit breaker” el cual se encarga de monitorear los fallos al realizar la petición, cuando se alcance cierto valor de fallos el objeto “circuit breaker” se activa y a partir de ese momento todas las peticiones que se realicen serán retornadas con un error hasta que después de un tiempo vuelva a intentarse realizar la petición si esta vuelve a fallar vuelve a empezar este proceso.

Según lo que hemos explicado anteriormente el “circuit breaker” puede encontrarse en tres estados:

1- Cerrado: Este es el estado inicial del objeto, en este estado se intentan realizar las peticiones.
2- Abierto: Este es el segundo estado y se llega a el desde el estado cerrado, esto se produce cuando se alcanza el umbral de fallos permitidos, en este estado no se permiten peticiones hasta que después de un tiempo se intenta realizar la petición.
3- Medio abierto: Si la petición del estado abierto es aceptada se vuelve al inicio, al estado cerrado, pero si esta es rechazada se retrocede al estado abierto.



¿Cuándo hay que usar “Circuit breaker”?

Este patrón se utiliza cuando queremos evitar que una dependencia de recursos, como puede ser una petición HTTP o la consulta a una base de datos, se sobrecargue. De manera que no siga recibiendo más peticiones si no va a ser capaz de satisfacerlas. Si nuestro objetivo es reducir el tiempo de espera del sistema en caso de sobrecarga este patrón puede ser la solución que buscamos. Es ideal cuando necesitamos un sistema resistente ante la sobrecarga.

Consideraciones previas

Antes de que implementes el patrón “Circuit breaker” en tus sistemas debes plantearte varias preguntas:

- **¿Tu sistema realmente va a mejorar?**
  - ¿Se va a producir sobrecarga?
  - ¿Reducir la carga va a solucionar el problema?
- **¿Eres capaz de implementarlo?**
  - ¿Puedes medir la carga que es capaz de manejar tu sistema?
  - ¿Eres capaz de ajustarlo correctamente?
- **¿Vas a poder mantenerlo?**
  - ¿Dispones del tiempo/dinero para medir la carga y volver a ajustar el “Circuit breaker” periódicamente?

Ajuste proactivo

El “Circuit breaker” es una solución fácil de implementar, mediante llamadas a una API, pero no debemos utilizarlo sin pensar, es muy importante realizar los ajustes correctos para sacar provecho de este patrón:

- Establecer un umbral aceptable para el cual se abre el circuito, ya sea por tasa de errores o por rendimiento.
- Definir el tiempo durante el cual el circuito se mantendrá abierto.
- Establecer otros disparadores que cierren el circuito cuando vuelva a estar operativo.
- Diferenciar estos ajustes en función del entorno, de los endpoints (componentes/dispositivos) y en caso de que no se traten igual, también de las peticiones.

Y lo más importante es que estos ajustes deben ser actualizados a medida que el sistema o la dependencia evolucionan.

Consecuencias de un “Circuit breaker” mal ajustado

1. Tu servicio proporciona un rendimiento menor del que es capaz de dar.
2. Estás alargando el tiempo de espera, incluso cuando el sub-sistema ya está operativo.
3. Es difícil entender cómo se están gestionando las peticiones.
4. Obtienes errores que no tendrías si no hubieses implementado el “Circuit breaker”.