CIRCUIT BREAKER

¿Qué es un Circuit Breaker?

Patrón que permite cerrar la comunicación con un determinado servicio cuando se detecta que está fallando.

Roles

- Cliente: intenta consumir los servicios del servicio (Supplier). Lo hace por medio del Circuit Breaker.
- Circuit Breaker: se encarga de la comunicación el servicio. Mantiene un registro del estado de salud del servicio.
- Servicio (Supplier): servicio con funcionalidad para ser consumida por el cliente.

Funcionamiento

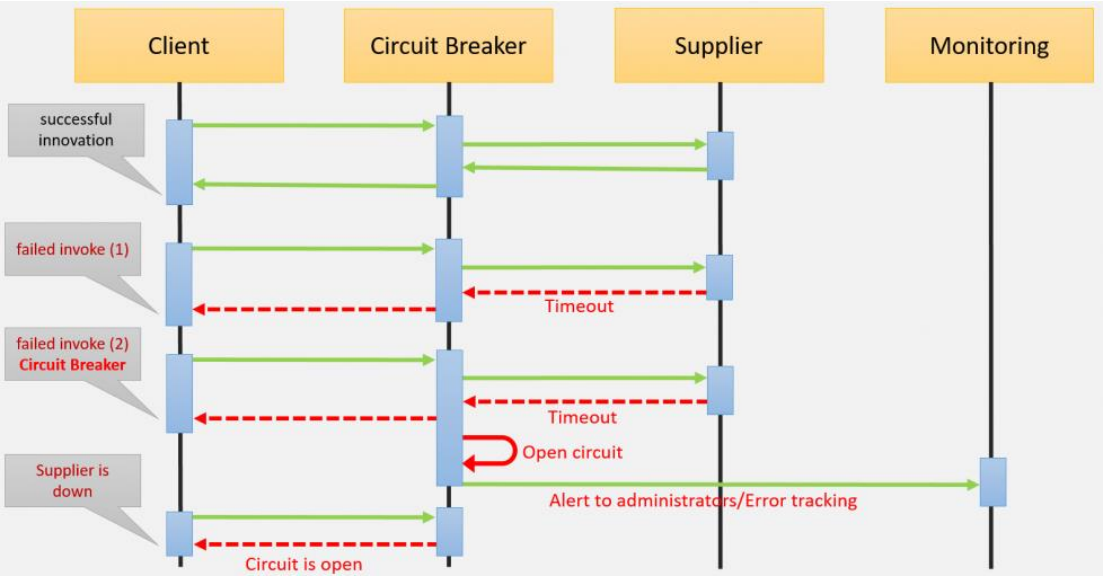
El cliente realizará llamadas al servicio por medio del Circuit Breaker. Este último determina si el servicio esta funcionando correctamente y redirecciona la petición al Supplier, el cual enviará la respuesta al cliente por medio del Circuit Breaker.

Si el Supplier devolviese un error o un Timeout, el Circuit Breaker tomaría nota del error. Pondría un contador a 1 de las veces que ha fallado y redireccionaría el error al cliente.

Dada una configuración, el Circuit Breaker puede determinar que el número máximo de errores simultáneos que puede tolerar sea 2.

Si nuevamente el Supplier volviese a dar error, el Circuit Breaker incrementaría el contador a 2 por lo que se abriría el circuito. Es muy recomendable tener algún elemento de monitoreo para notificar a los administradores en tiempo real sobre los errores producidos. Por último el error se enviaría al cliente.

A partir de este momento si el cliente envía peticiones al servicio, el Circuit Breaker directamente retornará el error al Cliente indiciándole que el Supplier está fuera de servicio (ha respondido con error un número determinado de veces).



En este momento, el circuito está abierto y será necesario cerrarlo una vez el problema se haya resuelto. Se debe configurar para ello cuánto tiempo debe esperar el Circuit Breaker antes de mandar una solicitud al Supplier.

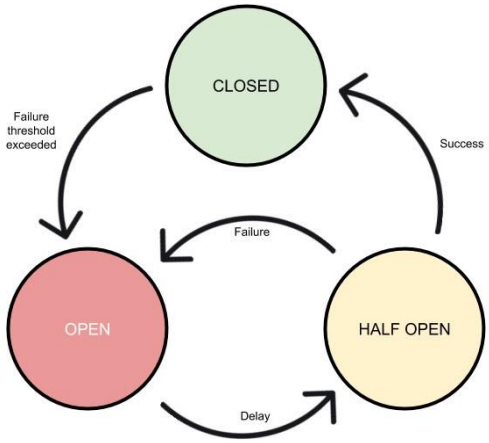
Si esperamos una hora desde que se abrió el circuito y enviamos una petición la cual retorna exitosamente, el Circuit Breaker cerrará el circuito.

Si la petición devuelve error entonces el tiempo de espera se reiniciará y habrá que esperar otra hora antes de mandar una nueva petición para comprobar la disponibilidad del servicio.

Ciclo de vida

Este patrón puede pasar por 3 estados diferentes:

- Closed: el servicio está respondiendo correctamente
- Open: el servicio está fallando y el Circuit Breaker regresa un error inedianto. Tras un tiempo de espera pasará a estado Half-Open
- Half-Open: puede recibir una pequeña cantidad de solicitudes para validar si el servicio está nuevamente activo. Si las solicitudes pasan exitosamente, se pasa a Closed; en caso contrario, se vuelve a Open.



Entornos Asíncronos

El patrón se usa de igual manera pero ahora empleando colas (queues). El Circuit Breaker en lugar de ejecutar directamente el servicio tendrá que depositarlo en una cola, de tal forma que el Supplier pueda ir procesándolos sin saturaciones y colapsos.

El Circuit Breaker podrá determinar el número de mensajes que tiene la cola y si este alcanza el umbral determinado se podrá determinar que el Supplier está fallando por lo que abriremos el circuito.

Los Circuit Breaker requieren un ajuste cuidadoso

Los Circuit Breaker son un ejemplo de técnicas de software de autorreparación. También son difíciles de hacer bien, por lo que no debemos usarlos sin pensar.

El patrón es extremadamente atractivo, puede evitar todo tipo de problemas que provienen de sistemas sobrecargados temporalmente. Parece que es algo fácil de implementar. Sin embargo, implementar un patrón de Circuit Breaker requiere un ajuste cuidadoso:

- Debe ajustar el umbral (tasa de error o rendimiento) en el que se abre el circuito.
- Debe ajustar el tiempo durante el cual el circuito permanece abierto y posiblemente ajustar otros activadores que permitan que el circuito se cierre.
- Necesita ajustar estas configuraciones por entorno y por endpoint como mínimo y, en algunos casos, posiblemente incluso por solicitud si las solicitudes al mismo endpoint no son todas iguales.

Sin embargo, lo más importante es que debe mantener estos ajustes a medida que evolucionan tanto su servicio como las dependencias subyacentes. Para ser más directo: el ajuste proactivo de los interruptores automáticos es fundamental.

Problemas de ajustar mal el patrón

Hay cuatro problemas que puede encontrar si no sintoniza correctamente el patrón con regularidad:

1- Su servicio proporciona menos rendimiento del que debería poder proporcionar.
2- Se alarga el tiempo de inactividad durante el período de tiempo en el que el circuito está abierto, incluso si el recurso subyacente se ha recuperado.
3- Se crean patrones de solicitud episódica que pueden resultar confusos para que los propietarios de servicios posteriores los comprendan.
4- Se le avisa por errores que no habrían ocurrido si no hubiera un Circuit Breaker.

Es decir, la aplicación ingenua de un interruptor automático puede causar fácilmente más problemas de los que resuelve.

Beneficios y razones para usar el patrón de Circuit Breaker

Estos son los principales motivos por los que interesa utilizar el Circuit Breaker:

- Falla rápido (mejores tiempos de respuesta y sin solicitudes colgantes).
- Sin una carga adicional innecesaria en el sistema / servicio que está fallando.
- Perfecto para implementar la lógica de reserva.
- Buena forma de realizar un seguimiento del estado del servicio para paneles y alarmas.
- Gracias a las bibliotecas es relativamente fácil de implementar.

Puntos a tener en cuenta

Si decidimos implementar el Circuit Breaker, aquí hay algunas cosas que puede considerar:

- No hay necesidad de reinventar nada. Probablemente haya bibliotecas para esto en su idioma. Opossum es un gran paquete npm para manejarlo.
- ¿Qué errores realmente desea contar para el umbral de error? Por ejemplo, es posible que no se desee disparar el Circuit Breaker debido a errores 404.
- Si bien el concepto general se traduce bien en software, los términos "cerrado" y "abierto" pueden interpretarse como lo contrario de lo que significan en este contexto.
- Considere dónde está mejor ubicado el Circuit Breaker. Por ejemplo, ¿necesita Circuit Breaker independientes en puntos finales específicos de una API o desea que todas las solicitudes a esa API compartan el mismo Circuit Breaker? Vale la pena señalar que lo último significa que si la API tiene uno o varios puntos finales que fallan con frecuencia, el Circuit Breaker podría dispararse a pesar de que otros puntos finales funcionen correctamente.

Bibliografía

https://martinfowler.com/bliki/CircuitBreaker.html

https://blog.newrelic.com/engineering/circuit-breaker-pattern/

https://medium.com/bonniernewstech/circuit-breaker-pattern-what-and-why-a17f8babbec0

https://www.oscarblancarteblog.com/2018/12/04/circuit-breaker-pattern/

Candela Bobes Junquera – UO269948

Diego Tomás Nosti - UO270497