Национальный исследовательский ядерный университет
«МИФИ» (Московский Инженерно–Физический Институт)
Кафедра №42 «Криптология и кибербезопасность»

ОТЧЁТ

Дополнительная работа:
«Circuit Breaker»

Группа Б21-525
Студент Р.Т. Мясников
Преподаватель М.А. Куприяшин

Москва 2024

Оглавление

1. Введение

Целью данной работы является разработка паттерна Circuit Breaker. Паттерн обеспечивает управление доступом к сервисам, предотвращая перегрузку системы при возникновении сбоев или недоступности сервиса.

2. Реализация паттерна

Для разработки Circuit Breaker была выбрана следующая схема работы:

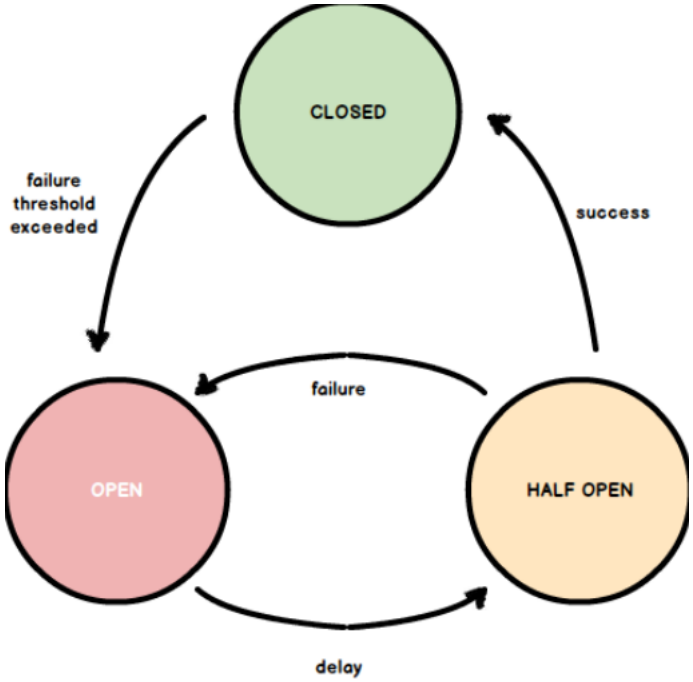

Рис. 1. Схема работы Circuit Breaker

Состояния Circuit Breaker

Паттерн Circuit Breaker включает в себя три состояния:

— `StateClose`: По умолчанию Circuit Breaker находится в этом состоянии, позволяя запросам проходить;
— `StateOpen`: Circuit Breaker переходит в это состояние, когда количество ошибок превышает установленный порог. В этом состоянии Circuit Breaker блокирует все запросы и отклоняет их, чтобы предотвратить перегрузку сервиса;
— `StateHalf`: Если Circuit Breaker находится в этом состоянии, он разрешает только одному запросу пройти, чтобы проверить, восстановился ли сервис или нет.

Параметры Circuit Breaker

Для настройки паттерна используется два параметра:

— `threshold`: Пороговое значение, указывающее количество ошибок, после которого Circuit Breaker переводится в состояние `StateOpen`;
— `timeout`: Время ожидания, в течение которого Circuit Breaker остается в состоянии `StateOpen`.

Методы Circuit Breaker

— `Execute(operation)`: Выполняет операцию, обернутую в Circuit Breaker. Если Circuit Breaker разрешает выполнение операции, она выполняется; в противном случае возвращается ошибка.

Логика работы Circuit Breaker

1. Перед выполнением операции проверяется текущее состояние Circuit Breaker;
2. Если Circuit Breaker находится в состоянии `StateOpen` и прошло достаточно времени с момента последней неудачной операции, он переводится в состояние `StateHalf`;
3. Если Circuit Breaker находится в состоянии `StateOpen` и не прошло достаточно времени, возвращается ошибка;
4. Выполнение операции;
5. После выполнения операции обновляется состояние Circuit Breaker в зависимости от результата операции и текущего состояния: `StateHalf` - в зависимости от ошибки работы сервиса статус меняется на `StateOpen` или `StateClose`; `StateClose` - если превышается число ошибок статус меняется на `StateOpen`.

3. Разработка паттерна

Был разработан Circuit Breaker на языке Go. Было реализовано поддержка параллельного выполнения операций, что позволяет обрабатывать паралелльные запросы, обеспечивая безопасное обновление внутреннего состояния механизма. Кроме того, благодаря поддержке параллелизма, существует возможность распределить нагрузку между рутинами, обеспечивая более равномерное и эффективное использование ресурсов.

Были разработаны модульные тесты, охватывающее как последовательное, так и параллельное выполнение операций. Для обеспечения качества и надежности паттерна были созданы тесты, которые проверяли его функциональность и корректность работы в различных сценариях использования: в тестах проверяется, что Circuit Breaker корректно переходит между состояниями в зависимости от результата выполнения операций, а также что он правильно обрабатывает ошибки и блокирует доступ при достижении порога ошибок.

4. Заключение

В результате был разработан механизм, который обеспечивает защиту от перегрузки системы при возникновении проблем с сервисом. Circuit Breaker позволяет эффективно управлять доступом к сервисам, предотвращая ненужные запросы в случае их недоступности или нестабильной работы. Это обеспечивает более надежное и стабильное функционирование приложения.

5. Приложение

Реализация: GitHub