УДК 004.42

РАЗРАБОТКА БИБЛИОТЕКИ ДЛЯ ОБЕСПЕЧЕНИЯ ОТКАЗОУСТОЙЧИВОСТИ РАСПРЕДЕЛЕННЫХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Е. С. Жаворонок

Белорусский государственный университет, Беларусь, Минск, evgeniyzhavoronok@gmail.com

В результате работы была реализована библиотека для обеспечения отказоустойчивости распределенных систем с использованием языка программирования Python. Описаны основные паттерны, использованные в библиотеке, принцип их работы и условия, при которых необходимо их использование. Описана также система, с помощью которой реализована возможность комбинации паттернов и их одновременного выполнения.

Ключевые слова: отказоустойчивость; распределенные системы; паттерны; Bulkhead; Circuit Breaker; Rate Limiter; Timeout; Fallback; Python.

DEVELOPMENT OF A LIBRARY FOR ENSURING FAULT TOLERANCE OF DISTRIBUTED SYSTEMS USING THE PYTHON PROGRAMMING LANGUAGE

Y. S. Zhavaronak

Belarussian state university, Belarus, Minsk, evgeniyzhavoronok@gmail.com

As a result of the work, a library for ensuring fault tolerance of distributed systems using the Python programming language was developed. The main patterns used in the library are described, along with their principles of operation and the conditions under which they should be used. Additionally, a system that enables the combination of patterns and their simultaneous execution is described.

Keywords: fault tolerance; distributed systems; patterns; Bulkhead; Circuit Breaker; Rate Limiter; Timeout; Fallback; Python.

Введение

Современные распределенные системы сталкиваются с множеством вызовов: частичные отказы компонентов, сетевые задержки, перегрузки серверов и асинхронные взаимодействия. Обеспечение отказоустойчивости таких систем требует реализации специализированных паттернов проектирования, которые минимизируют каскадные сбои и повышают надеж-

ность. Система называется отказоустойчивой, если она способна функционировать даже в том случае, если какая-то ее часть работает некорректно [1].

Существующие решения для Python либо фокусируются на отдельных паттернах, не предлагая комплексного подхода к их комбинированию (tenacity, pybreaker), либо реализуют только небольшое количество паттернов (pyfailsafe), либо работают только для библиотеки asyncio (hyx).

Цель работы — разработка библиотеки, предоставляющей гибкий инструментарий для создания отказоустойчивых конвейеров обработки запросов. В библиотеке реализованы следующие паттерны: Bulkhead, Circuit Breaker, Rate Limiter, Timeout, Fallback. Библиотека позволяет одновременно использовать несколько стратегий обработки сбоев через класс Pipeline.

Актуальность исследования обусловлена ростом популярности Python в микросервисной архитектуре и data science, где требования к отказоустойчивости систем критически важны.

Описание реализованных паттернов и детали реализации

Хотя паттерны могут использоваться по отдельности, основой библиотеки является класс Pipeline. Он позволяет комбинировать несколько паттернов для их одновременного выполнения. Сначала в том порядке, котором они были добавлены, выполняются паттерны, которые проверяют, может ли функция быть вызвана в данный момент: Bulkhead, Circuit Breaker и Rate Limiter. После этого начинается исполнение функции. При этом, если в объекте Pipeline есть паттерн Timeout, то функция вызывается через него. В случае если один из паттернов заблокировал выполнение функции или произошла ошибка, вызывается объект, переданный в паттерн Fallback.

Паттерн Bulkhead изолирует ресурсы системы, предотвращая каскадные сбои – ситуации, когда сбой в одном компоненте приводит к сбоям в других компонентах, зависящих от него. Данный паттерн выделяет определённое количество одновременных вызовов, которые доступны в данном контексте. Таким образом, даже если один из компонентов системы блокирует ресурсы, то он сможет заблокировать только то количество ресурсов, которое было для него выделено. При этом остальные компоненты, которые делят с ним один пул ресурсов, смогут продолжать работу, так как пул ресурсов не будет полностью истощен.

Паттерн Circuit Breaker также позволяет избежать каскадных сбоев, когда один из компонентов системы выходит из строя [2]. Паттерн имеет три состояния: закрытое, открытое и полуоткрытое. В закрытом состоянии запросы проходят как обычно и ведётся подсчет ошибок за опреде-

ленный промежуток времени. В открытом состоянии все запросы блокируются, и система не пытается выполнить операции. В полуоткрытом состоянии разрешено выполнение определенного количества операций до первого сбоя. Изначально паттерн находится в закрытом состоянии и, когда количество ошибок за определенный промежуток времени переходит пороговое значение, паттерн переходит в открытое состояние. В открытом состоянии он находится заданное количество времени, после чего переходит в полуоткрытое состояние. Если определенное количество операций в полуоткрытом состоянии выполняется успешно, паттерн переходит в закрытое состояние, иначе возвращается в открытое [3].

Паттерн Rate Limiter контролирует нагрузку на систему, предотвращая слишком частые запросы. При этом слишком большое количество запросов может означать не только действительно большую нагрузку от пользователей, но и то, что в системе произошел сбой и запросы, которые завершаются неудачно, сразу же вызываются заново. Паттерн Rate Limiter позволяет ограничить количество запросов в определённый промежуток времени. Паттерн Timeout прерывает слишком долго выполняющиеся операции для освобождения ресурсов. Если операция не закончилась за заданное время, она принудительно останавливается с помощью исключения.

Паттерн Fallback обеспечивает альтернативный сценарий поведения в случае, если результат основной функции был неудовлетворительным. Если основная функция закончилась с исключением или её возвращаемое значение не было допустимым, вызывается объект, переданный в Fallback.

Заключение

Созданная библиотека предоставляет разработчикам возможность использования паттернов отказоустойчивости в языке программирования Python для обработки запросов, позволяя их эффективно комбинировать. Это значительно упрощает процесс разработки надежных систем, которые могут справляться с частичными сбоями и сетевыми проблемами. Библиотека является расширяемой и кроме локального хранения состояния позволяет наследоваться от классов, реализующих паттерны, и создавать классы, которые будут хранить состояние в базах данных.

Библиографические ссылки

1. *Ганмер Р.* Patterns for Fault Tolerant Software / 1-е изд. Wiley, 2007. 23 с.

2. Circuit Breaker [Электронный ресурс]. URL: https://martinfowler.com/bliki/CircuitBreaker.html (дата обращения: 21.03.2025).

3. *Молчанов Г. И.* Circuit Breaker в системах на основе микросервисной архитектуры / Advanced Information Systems. 2018. Т. 2. № 4. С. 74–77.