УДК 004.75

РАЗРАБОТКА АРХИТЕКТУРНОГО ПАТТЕРНА ДЛЯ ОТКАЗОУСТОЙЧИВОГО ВЗАИМОДЕЙСТВИЯ ВЫСОКОНАГРУЖЕННЫХ ПРИЛОЖЕНИЙ С БРОКЕРАМИ СООБЩЕНИЙ

Агеев А.Е. (ИТМО)
Научный руководитель – инженер, Петухов П.Н. (ИТМО)

Введение. В современном мире высоконагруженные IT-системы требуют высокой отказоустойчивости, так как сбои могут привести к значительным финансовым потерям и снижению качества предоставляемых услуг. Основные проблемы включают единую точку отказа, перегрузку систем сообщений и сложность диагностики. Взаимодействие с брокерами сообщений также может вызывать сбои, такие как потеря сообщений, повторная доставка и узкие места в обработке очередей [1]. Эти проблемы могут негативно повлиять на доступность и надежность системы.

В данной работе рассматривается подход к минимизации сбоев при работе с брокерами сообщений путем использования отказоустойчивого паттерна, который сочетает в себе сохранение сообщений в базе данных перед отправкой (Outbox), управление ошибками через отбрасывание, паузу или повторную попытку (Discard, Pause and Retry) и контроль количества запросов к серверу за определённый период (Rate Limiting) [2]. Данный подход позволяет предотвратить перегрузку брокера сообщений, обеспечивая надежность и стабильность работы сервисов.

Основная часть. Одной из ключевых проблем в высоконагруженных системах является ситуация, когда некорректно обработанные сообщения начинают накапливаться в очереди, мешая нормальному прохождению валидных сообщений [3]. Это может привести к перегрузке системы, увеличению задержек обработки и снижению надежности сервиса. В таких случаях традиционные механизмы повторной отправки сообщений могут не справляться с проблемой, что требует применения более гибких решений [4].

Для решения данной проблемы был разработан алгоритм, позволяющий идентифицировать и изолировать проблемные сообщения, помещая их в отдельную очередь [2]. Такой подход предотвращает блокировку основной очереди и позволяет обрабатывать валидные сообщения без задержек. В дальнейшем отложенные сообщения могут быть обработаны повторно или подвергнуты дополнительному анализу для исправления ошибок.

На основе предложенного алгоритма был разработан программный прототип. В качестве технологий использованы Kotlin, Spring Boot, Kafka и PostgreSQL. Разработанный сервис позволяет управлять потоком сообщений, обеспечивая их надежную доставку и минимизируя негативные последствия сбоев. В системе реализован механизм динамического мониторинга очередей и автоматического переноса проблемных сообщений в отдельное хранилище.

Для оценки эффективности предложенного решения были проведены эксперименты. Система подвергалась нагрузочному тестированию с использованием Gatling, а мониторинг ее работы осуществлялся с помощью Prometheus и Grafana [5]. Результаты тестирования показали, что использование предложенного паттерна позволило существенно снизить нагрузку на брокер сообщений, улучшить обработку ошибок и предотвратить потери сообщений даже в условиях высокой нагрузки

Выводы. В данной работе был предложен архитектурный паттерн, направленный на обеспечение отказоустойчивого взаимодействия высоконагруженных приложений с брокерами сообщений. Реализация предложенного подхода позволила добиться изоляции проблемных сообщений, предотвращая их влияние на основной поток обработки данных. Проведенные эксперименты подтвердили эффективность предложенного решения, обеспечив снижение нагрузки на брокеры сообщений и улучшение пропускной способности системы. Будущие исследования могут быть направлены на интеграцию данного паттерна с

различными средами оркестрации контейнеров, такими как Kubernetes, для дальнейшего повышения устойчивости и автоматического управления сбоями.

Список использованных источников:

1. Seymour M. Mastering Kafka Streams and ksqlDB. – O'Reilly Media, 2021.
2. Ivaki N., Laranjeiro N., Araujo F. Towards designing reliable messaging patterns //2016 IEEE 15th International Symposium on Network Computing and Applications (NCA). – IEEE, 2016. – С. 204-207.
3. Michelson B. M. Event-driven architecture overview //Patricia Seybold Group. – 2006. – Т. 2. – №. 12. – С. 10-1571
4. Baj B. et al. (ed.). Distributed Intelligent Circuits and Systems. – World Scientific, 2024. – Т. 3.
5. Sridharan C. Distributed systems observability: a guide to building robust systems. – O'Reilly Media, 2018