УДК 004.4

РАЗРАБОТКА РАСПРЕДЕЛЕННОЙ ОЧЕРЕДИ ЗАДАЧ ДЛЯ УДАЛЕННОГО ВЫЗОВА ПРОЦЕДУР С ПОДДЕРЖКОЙ АСИНХРОННОГО ПРОГРАММИРОВАНИЯ

Котовщиков А. Р.[1]
Научный руководитель – инженер Мигулаева Т. А.[1]
[1]Университет ИТМО
ykt_andrey@mail.ru

Введение

В современной разработке ПО микросервисная архитектура становится все более распространенным решением [1]. Данная тенденция способствует повышению спроса на брокеры сообщений и, как следствие, на системы управления очередями задач, которые предоставляют удобную и надежную абстракцию над ними. Однако, анализ существующих решений выявил ряд существенных ограничений в контексте современных парадигм разработки. Во-первых, наблюдается отсутствие поддержки асинхронного программирования [2]: многие системы были ориентированы на I/O нагрузку, что приводит к необходимости использования специальных оберток, снижающих производительность, и к неэффективному использованию ресурсов системы. Во-вторых, для большинства популярных решений характерны архитектурные недостатки, такие как использование глобального состояния приложения (так называемый «global app» антипаттерн) [3], что препятствует простоте тестирования и нарушает принципы чистой архитектуры. Одной из ключевых проблем является отсутствие обособленного RPC-клиента, что приводит к жесткой связности кодовой базы микросервисов: отправитель и получатель вынуждены использовать общие модули, что полностью противоречит принципам слабой связанности в микросервисной архитектуре. Таким образом, существует потребность в новом решении, которое сочетало бы надежность и функциональность существующих аналогов, но при этом было бы лишено их ключевых архитектурных недостатков.

Основная часть

Новая распределенная очередь обеспечивает поддержку нескольких брокеров сообщений (RabbitMQ и NATS/JetStream), механизмы вертикального (увеличение числа воркеров) и горизонтального масштабирования (запуск инстансов на различных хостах) [4], а также предлагает объектно-ориентированный подход к описанию логики задач [5] с поддержкой асинхронного программирования и обособленный RPC-клиент для взаимодействия. Новый инструмент нивелирует недостатки существующих систем, предоставляя унифицированный интерфейс для работы с брокерами, что снижает порог вхождения, минимизирует объем самописного кода и сопутствующие ошибки, позволяя разработчикам абстрагироваться от особенностей настройки и функционирования конкретных брокеров и сфокусироваться на прикладных бизнес-задачах.

Выводы

Разработанный в ходе исследования инструмент представляет собой надежное решение для асинхронного взаимодействия сервисов, позволяющее существенно сократить временные затраты разработчиков. Универсальность предложенной очереди обеспечивает ее применение как для решения стандартных задач коммуникации в микросервисной архитектуре, так и для реализации более специфичных сценариев, включая поддержание саг (повествований) для распределенных транзакций и построение событийно-ориентированных систем.

Литература

1. Richardson C., Microservices Patterns: With examples in Java. – 2018.
2. Celery – Distributed Task Queue [Электронный ресурс]. – Режим доступа: https://docs.celeryq.dev/ (дата обращения: 20.02.2026).
3. Dramatiq: Background tasks [Электронный ресурс]. – Режим доступа: https://dramatiq.io/index.html (дата обращения: 20.02.2026).
4. Kleppmann M., Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. – 2017.
5. Robert C. Martin, Clean Architecture: A Craftsman's Guide to Software Structure and Design. – 2017.