ПРИМЕНЕНИЕ ПАТТЕРНА FAN-OUT ДЛЯ ОБНОВЛЕНИЯ ЛЕНТЫ В ПРИЛОЖЕНИИ ДЛЯ ВЗАИМОДЕЙСТВИЯ В ТЕМАТИЧЕСКИХ СООБЩЕСТВАХ

Алексеева К.А.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Петюкевич Н.С. – ст. преподаватель

Современные платформы для тематических сообществ должны обеспечивать быстрое обновление контента, чтобы поддерживать интерес и связь между участниками сообществ. Обычные способы обработки данных часто уступают под напором активности, создавая задержки в доставке публикаций и снижая динамику общения. Разработанное веб-приложение использует паттерн Fan-out, который позволяет платформе оперативно отображать новые публикации в пользовательских лентах.

В последние годы наблюдается стремительный рост интереса к тематическим сообществам. Пользователи всё чаще объединяются вокруг общих интересов – будь то кулинария, путешествия или технологии – и предъявляют высокие ожидания к цифровым платформам, обеспечивающим их взаимодействие. Одним из ключевых компонентов таких приложений является персонализированная лента, призванная оперативно отображать новые публикации в рамках сообществ, на которые подписан пользователь.

Традиционные подходы к формированию ленты, основанные на периодическом опросе базы данных, сталкиваются с рядом существенных ограничений. Они приводят к задержкам в отображении актуального контента и создают значительную нагрузку на сервер, особенно в условиях высокой активности пользователей. В результате ухудшается пользовательский опыт, негативно сказываясь на удержании аудитории, для которой своевременность информации имеет первостепенное значение.

Проблемы становятся особенно заметными при масштабировании. Когда число публикаций и пользователей растет, частые обращения к базе данных для чтения публикаций и формирования лент становятся неэффективными, так как требуют значительных ресурсов и увеличивают время отклика.

Для решения этих задач в разработанном приложении применен паттерн Fan-out, который обеспечивает доставку новой публикации в ленты всех подписчиков сразу после ее создания. Это позволяет обновлять контент асинхронно, минимизируя обращения к базе данных и снижая нагрузку на систему [1].

Приложение построено на микросервисной архитектуре с использованием современных технологий. Серверная часть реализована на ASP.NET Core, клиентская – на Angular. PostgreSQL выступает основным хранилищем для долговременных данных. Redis применяется для кэширования лент пользователей, так как обеспечивает быстрый доступ к данным в памяти [2].

Когда пользователь создает новую публикацию, сервис публикаций отправляет событие в Apache Kafka, распределенную платформу для потоковой обработки событий. Kafka выступает в роли посредника, обеспечивая надежную и масштабируемую передачу событий заинтересованным сервисам.

Сервис обработки событий, подписанный на соответствующие топики в Kafka, получает это событие и начинает процесс обновления лент. Для определения, какие пользователи должны увидеть публикацию, сервис использует данные о подписках. Их он получает из сервиса сообществ через gRPC запросы.

Сервис обработки событий извлекает список пользователей, связанных с сообществом публикации, и для каждого такого пользователя добавляет публикацию в их ленту, хранящуюся в Redis. Этот и является сутью паттерна Fan-out: одно событие (новая публикация) распространяется на множество назначений (ленты пользователей).

Паттерн Fan-out в сочетании с Kafka и Redis позволяет быстро обновлять ленты даже при высокой нагрузке. Это обеспечивает отзывчивость системы, способной обрабатывать тысячи событий в секунду. Распределенная архитектура гарантирует стабильность и масштабируемость.

В итоге применение паттерна Fan-out и микросервисной архитектуры эффективно решает задачу быстрой доставки контента в тематических сообществах. Решение готово к росту аудитории и создает основу для дальнейшего развития платформы, поддерживая высокую производительность и вовлеченность пользователей.

Список использованных источников:

1. The Architecture Twitter Uses to Deal with 150M Active Users, 300K QPS, a 22 MB/S Firehose, and Send Tweets in Under 5 Seconds [Электронный ресурс]. – Режим доступа: https://highscalability.com/the-architecture-twitter-uses-to-deal-with-150m-active-users.

2. Building Scalable News Feed Applications Using Redis and Cassandra [Электронный ресурс]. – Режим доступа: https://thenewstack.io/building-scalable-news-feed-applications-using-redis-and-cassandra.