УДК 004.9

СРАВНЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ СТРАТЕГИЙ ЗАПРОСОВ СВЯЗАННЫХ ДАННЫХ В МИКРОСЕРВИСНОЙ АРХИТЕКТУРЕ

Кондрашов Е. Ю. (Университет ИТМО, Санкт-Петербург)
Научный руководитель – кандидат технических наук, старший научный сотрудник Ходненко И. В.
(Университет ИТМО, Санкт-Петербург)

Введение. Часто используемым подходом к хранению данных в микросервисной архитектуре является подход database-per-service (и его вариации, schema-per-service, private-tables-per-service) [1]. Для реализации этого подхода микросервисы должны взаимодействовать с данными других только через API, а не обращаясь к их базам данных напрямую. При использовании описанного подхода возникают проблемы, которые не присущи монолитной архитектуре. В том числе, при необходимости отображать связанные данные из разных микросервисов.

Несмотря на то, что существуют подходы к решению этой проблемы, в случае необходимости поддерживать фильтрацию и пагинацию по связанным данным из разных хранилищ нет источников, показывающих, как выбор того или иного подхода повлияет на производительность системы. Для сравнения производительности были проведены эксперименты с помощью нагрузочного тестирования. По результатам тестирования были даны рекомендации насчёт предпочтительного подхода.

Основная часть. Основные подходы, это, во-первых, API Composition, когда связанные данные запрашиваются от других сервисов через их API [1]. Полный ответ формируется бэкендом одного из микросервисов, связывая данные по ключам.

Во-вторых, использование общей БД, из который берутся все данные, необходимые для обработки запроса. Этот подход требует поддерживать копию на чтение данных из других сервисов. Для этого можно обрабатывать эвенты от этих сервисов. Можно использовать репликацию средствами СУБД [2]. При использовании разных технологий хранения данных, для поддержания реплик могут применяться ETL-инструменты, например, SymmetricDS.

В-третьих, использование федеративного доступа к данным, когда сервер БД сам запрашивает их от внешнего источника, например, через PostgreSQL Foreign Data Wrapper [3] или Oracle Database Gateway. Описание применения этого варианта не встречается в изученных источниках, но он вполне подходит для задачи.

Для сравнения производительности была создана демонстрационная система из трёх микросервисов A, B и C и синтезированы данные. Демонстрационная система основана на системе управления обучением и хранит данные по студентам, их курсам и оцениваемым заданиям. Для тестирования использовались три хоста, на каждом из которых было размещено по одному микросервису. В нагрузочном тестировании количество клиентов увеличивалось от 0 до 2000 с запуском 50 новых в секунду. Клиенты выполняли списочный GET запрос к микросервису A, включающий связанные данные из микросервисов B и C. Кроме того, применялась фильтрация, её значение случайно выбиралось из числа всех существующих значений в базах данных. Также применялась пагинация, выбирая случайную страницу в пределах существующих. Собирались такие ключевые метрики, как количество обработанных запросов в секунду и среднее время ответа. Сравнивались конфигурации, когда в формировании ответа участвуют 2 и 3 микросервиса.

API Composition показал наихудший результат, в первом сценарии в пике в обрабатывая запросы в 29 раз медленнее, чем Foreign Data Wrapper. Foreign Data Wrapper показал наибольшую производительность в сценарии 1 и сценарии 2 с двумя сервисами. Использование общей БД показало близкий к нему результат, а в сценарии 2 с тремя сервисами оказалось наиболее производительным.

Выводы. В результате проведённых экспериментов сделан вывод, что API Composition не рекомендуется использовать, если требуется отображать связанные данные из разных микросервисов с фильтрацией и пагинацией. Это важно, так как в нескольких источниках он рекомендуется как подход, который должен использоваться для отображения связанных данных всегда, когда это возможно [1]. Общая БД и федеративный доступ к данным обладают хорошей и схожей производительностью, поэтому при выборе стоит рассматривать другие их плюсы и минусы для конкретной системы.

Тот факт, что при использовании общей БД производительность не зависит от других микросервисов и наоборот, нагрузка не влияет на них, является весомым аргументом за использование этого подхода по умолчанию.

Список использованных источников:

1. Richardson C. Microservices patterns: with examples in Java. – Simon and Schuster, 2018.
2. Wolff E. Microservices: flexible software architecture. – Addison-Wesley Professional, 2016.
3. PostgreSQL Documentation: postgres_fdw — access data stored in external PostgreSQL servers [Электронный ресурс]. — URL: https://www.postgresql.org/docs/current/postgres-fdw.html (дата обращения 11.02.2025).