УДК 004.031.42

ИССЛЕДОВАНИЕ МОДЕЛИ МИКРОСЕРВИСОВ СЕТЕВЫХ ПРИЛОЖЕНИЙ

Дунаевский М.А.
к.т.н., доцент каф. КИСС Мартынюк А.Н
Одесский Национальный Политехнический Университет, УКРАИНА

АННОТАЦИЯ. Исследована модель микросервисов. Определены преимущества и недостатки архитектуры. Предложено решение увеличения производительности и уменьшения задержек при обмене данными.

Введение. Микросервисы - это новая тенденция, которая быстро развивается. Микросервисы - это небольшие, автономные, совместно работающие сервисы [1]. Архитектура микросервиса - становится все более популярной, так как может ускорить гибкую разработку программного обеспечения, развертывание и эксплуатации.

Цель работы. Проведение исследования модели микросервисов и выявления оптимального протокола обмена данными для повышения производительности и снижения времени передачи данных.

Основная часть работы. В распределенных системы необходимо учитывать критерии распределенного вычисления при разработке приложений, такие как: Согласованность данных(*Consistency)* – во всех узлах распределённой системы в один момент времени все данные одинаковы и не противоречат друг другу; Доступность(*Availability*) – на каждый запрос к узлу распределённой системы приходит ожидаемый ответ, но нет гарантии, что ответы в один момент времени будут совпадать; Устойчивость к разделению(*Partition tolerance*) —разделение узлов распределённой системы на несколько изолированных частей не приводит к неожидаемой работе системы. В таблице 1 приведено сравнение критериев выполняемых в монолитной и микросервисной архитектуре.

Таблица 1 – Сравнительная характеристика архитектур по *CAP*

Технология	Согласованность данных	Доступность	Устойчивость к разделению
Монолитная	+	-	-
Микросервисы	-	+	+

Микросервисы обеспечивают такие преимущества, как: Сильные границы модулей – микросервисы укрепляют модульную структуру; Независимое развертывание – простые службы проще развертывать, и поскольку они автономны, реже возникают сбои системы; Разнообразие технологий – с помощью микросервисов можно смешивать несколько языков, разработки и технологии хранения данных.

При этом они не лишены недостатков: Распространение – распределенные системы сложнее программировать, поскольку удаленные вызовы медленны и всегда подвержены риску отказа; Конечная согласованность – поддержание сильной согласованности чрезвычайно сложно для распределенной системы, что означает, что каждый человек должен управлять конечной последовательностью [2].

Микросервисы используют распределенную систему для повышения модульности, но при этом множество удаленных вызовов добавляет характеристики задержки.

Для повышения производительности и уменьшения задержек возможна замена механизмов обмена данными. Чтобы проверить этого был построен микросервис на nodejs, отличающийся возможностью обмена данными по механизмам *Rest (Http) и gRPC*.

Протокол *Http* – это первоначальная концепция, которую реализуют в большинстве сервисах. Самая большая проблема – это потенциальные задержки для удаленных вызовов. Для преодоления проблем задержек используется механизм обмена данными gRPC.

gRPC, HTTP / 2 RPC Framework для микросервисов. *Google* отмечает *gRPC* «Высокая пропускная способность и эффективность использования процессора, низкие задержки для

создания широко распределенных систем, которые охватывают центры обработки данных, а также мощные мобильные приложения, связь в режиме реального времени, устройства IoT и API»[3]. На рисунке 1 изображена схема обмена данными между клиентом и сервисами в построенном микросервисе.

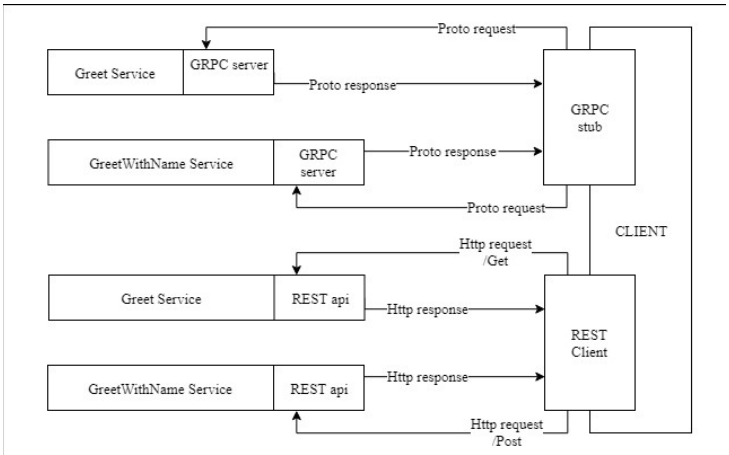

Рис. 1 – Схема обмена данными микросервисов и клиентов

После построения микросервиса был разработан клиент для сбора информации о прозводительности при обмене данными по заданным протоколам. Полученные данные отображены в таблице 2.

Таблица 2 – Сравнительная характеристика производительности

Механизм	Операций в секунду
Rest (Http)	450
gRPC	3000

По полученным данным можно увидеть, что разница в производительнсоти существенна и что протокол *gRPC* имеет большую скорость обработки запросов, чем *Http*, но при этом имеет более сложный процесс построения коммуникаций между сервисами и клиентом.

Выводы. В работе исследована модель микросервисов, которая имеет как преимущества так и недостатки. В качестве недостатка было отмечено добавление характеристик задержки при множестве удаленных вызовов. Для преодоления проблемы возможно использование других механизмов обмена данными, тем самым снижая задержки и увеличивая производительность. Для проверки этого был построен микросервис содержащий два разных механизма.

Исследование показало, что за счет перехода на другой механизм обмена данными возможно достичь увеличения производительности. а следовательно и сокращение времени передачи данных и возможных задержек. В построенном микросервисе за счет использования протокола *gRPC* вместо *Http* удалось увеличить производительность почти в 6 раз. Для получения результатов приближенных к реальным условиям необходимо построить полноценную систему, включающую в себя больше микросервисов, которые имеют более сложную логику работы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ньюмен С. Создание микросервисов. — СПб.: Питер, 2016. — 304 с.: ил. — (Серия «Бестселлеры O'Reilly»). ISBN 978-5-496-02011-4

2. Martin Fowler. Microservice Trade-Offs [Електронний ресурс]. – Режим доступу: URL: https://martinfowler.com/articles/microservice-trade-offs.html

3. Asia Pacific Journal of Contemporary Education and Communication Technology (APJCECT) ISBN: 978 0 9943656 82; ISSN: 2205-6181 Year: 2017, Volume: 3, Issue: 1 [Електронний ресурс]. – Режим доступу: URL: https://apiar.org.au/wp-content/uploads/2017/02/34_APJCECT_Feb_BRR7105_ICT-367-372.pdf