РАЗРАБОТКА FAAS ПЛАТФОРМЫ

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

Намакарский А.А.

Луцик Ю.А. – к.т.н.доцент

С развитием технологий архитектура серверной части приложений изменятся в сторону большей гибкости. Огромную популярность имеет конвейерная архитектура. Но не смотря на все ее плюсы, такие как слабая связность компонентов, возможность довольно быстрого внесения изменений в обработчики, отличную горизонтальную масштабируемость, у этого архитектурного паттерна есть один большой недостаток – сложность настройки и поддержки серверного окружения. Существуют десятки инструментов для решения этой задачи, однако, они требую высокой квалификации системных администраторов. Убрав необходимость поддержи серверного окружения можно существенно снизить затраты на разработку проектов, требующих наличия серверной части.

FaaS (Function as a service) – дословно переводится как “функция как сервис”, архитектурный паттерн относящийся к классу бессерверных архитектур. Идея данного архитектурного паттерна заключается в следующем: серверная часть приложения разбивается на набор функций, не хранящих состояния между вызовами, то есть результат выполнения функции не должен зависеть от состояния памяти сервера и локальной файловой системы, а зависит только лишь от переданных параметров. При наличии таких ограничений горизонтальное масштабирование представляется довольно простой задачей. Масштабирование выполняется провайдером для поддержания требуемого уровня производительности, в данном случае провайдером является FaaS платформа.

Принцип организации структурных компонентов серверной части при данном архитектурном походе показан на рисунке 1:

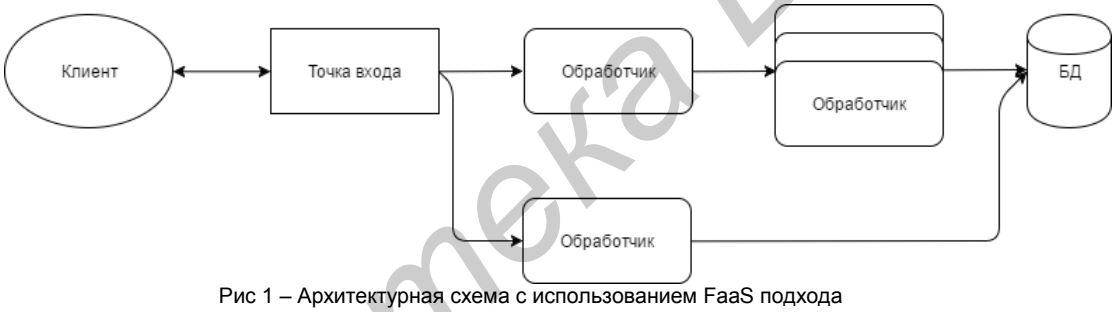

Рис 1 – Архитектурная схема с использованием FaaS подхода

На схеме видно, что для клиентской части приложения все выглядит так же, как и в случае с обычной клиент серверной архитектурой, что достигается с помощью единой точки входа, которая является подобием роутера. Данное решение позволит не вносить изменения в клиентскую часть, если она уже существует.

Серверная часть обладает некоторыми особенностями:

a) Отсутствует необходимость в настройке и обслуживании серверов
b) Отсутствует необходимость в использовании фреймворков, в данном случае обработчик будет простейшей программой, которая должна иметь определенный интерфейс. Это ведет к слабой связности компонентов.
c) Так как отсутствует серверная часть приложения, доставка изменений представляет собой простейшую задачу по загрузке кода. Например, в виде ZIP-архива
d) Задача горизонтального масштабирования полностью ложится на плечи FaaS-платформы и от разработчиков не требуется никакой конфигурации.
e) Функции вызывается с помощью событий, определенных провайдером, например, запрос на определенный пользователем URL, по таймеру и так далее.

Однако, данный подход также не лишен недостатков. При запуске функции-обработчика тратится дополнительное время на запуск изолированной среды с пользовательским кодом. Также существует необходимость ограничения выполнения функции-обработчика по времени, на случай если внутри этой функции произошло зацикливание. Также наличие большого количества функций-обработчиков может добавить трудностей при поддержке приложений. Данная проблем может быть частично решена с использованием гибридной архитектуры, в которой некоторая часть проекта будет реализована с применением FaaS подхода.

Данный подход позволит существенно снизить затраты на разработку малых и средних проектов, в которых необходимо наличие серверной части.

Список использованных источников:

1. Asif Qumer Gill, Adaptive Cloud Enterprise Architecture
2. Wilder B., Cloud Architecture Patterns. / B.Windler: Boston, 2012 – 182 с.