Эффективность реализации CRUD операций с использованием бессерверных вычислений

Шибаев Павел Павлович

Студент

Факультет ВМК МГУ имени М. В. Ломоносова, Москва, Россия

E-mail: shiba.p@ya.ru

Научный руководитель *— Колосов Алексей Михайлович*

Бессерверные вычисления представляют собой модель облачных вычислений, при которой ресурсы для вычислений выделяются по требованию, автоматически, без прямого участия клиента. Один из шаблонов построения бессерверной архитектуры — модель "функция как услуга"(Function as a Service, FaaS), которая предполагает размещение экземпляра управляющего кода или исполняемого файла в облаке. При поступлении запроса на выполнение происходит активация функции с выделением необходимых для выполнения ресурсов; по завершении происходит их высвобождение, что позволяет повысить утилизацию ресурса [1]. Несмотря на то, что такой подход позволяет разработчику абстрагироваться от инфраструктуры и сфокусироваться на написании кода, а также помогает рационально использовать ресурсы облачного сервиса, концептуально он мало чем отличается от широко распространившейся практики контейнеризации приложений. Облачная функция выполняется в контейнере, только теперь его конфигурирует не сам разработчик, а поставщик (вендор) облачных услуг [1]. При этом возникает проблема т.н. "холодного"старта — первичного запуска такого контейнера, что может занять порядка секунды. Впоследствии контейнер может быть многократно переиспользован (т.н. "горячий"старт), что позволяет амортизировать временные затраты на первый запуск.

Для современных web-приложений характерно широкое использование баз данных, в частности, особенно часто осуществляются разнообразные CRUD-операции (Create, Read, Update, Delete - добавление, чтение, обновление и удаление записей в базе данных). От приложений требуется высокая степень параллелизма, а также возможность асинхронного выполнения CRUD-запросов. В настоящий момент такой стиль программирования поддерживают многие языки и фреймворки (Golang, Rust, Node.js ...). При этом зачастую для каждого типа запросов пишется небольшая асинхронная функция-обработчик. Это делает возможным размещение функций-

обработчиков в облаке, что означает *перенос параллелизма на уровень инфраструктуры*.

Проведен эксперимент по сравнению классической реализации CRUD-запросов через приложение на сервере и реализации через облачные функции. В качестве языка программирования используется Golang (компилируемый язык, что в теории уменьшает время "холодного"старта), в качестве СУБД используется MongoDB (позволяет хранить json-подобные документы, не требуя размещения в функциях SQL-запросов или использования ORM-библиотек). Реализованы функции-обработчики для создания, чтения, обновления и удаления (CRUD) из базы данных [2].

Функции-обработчики размещены в пяти локациях: на арендуемой виртуальной машине в виде классического асинхронного приложения на основе http-роутера gorilla/mux, в виде облачных функций у четырех наиболее популярных поставщиков бессерверных вычислений Amazon Web Services, Google Cloud Platform, Microsoft Azure и IBM Cloud [3]. В качестве региона размещения ресурсов во всех случаях выбран Франкфурт, ФРГ. Выполнены замеры производительности в каждом из случаев для каждой операции. При запуске тестов создается 100 параллельно работающих клиентов, каждый из которых последовательно посылает 100 запросов. Результаты представлены в виде коробчатых диаграмм (boxplot) на Рис. 1 и Рис 2. (aws соответствует Amazon Web Service, gcp - Google Cloud Platform, az - Microsoft Azure, ibm - IBM Cloud, vm - виртуальной машине). Рис. 1 соответствует измерению времени от посылки клиентом запроса до получения ответа. Рис. 2 соответствует тому же времени, но за вычетом круговой задержки при подключении к облачному сервису.

В некоторых случаях производительность облачных функций сравнима с производительностью виртуальной машины. Кроме этого, разработка и тестирование простых облачных функций занимает гораздо меньше времени, нежели написание и отладка полноценного приложения. Наконец, и Microsoft Azure, и IBM Cloud предоставляют каждый месяц бесплатный грант на вычисления. Благодаря этому, например, сервис, к которому в месяц делается 1 млн обращений, при каждом из которых облачная функция работает примерно 1 с, используя 256 Мб RAM, будет бесплатным на обеих платформах [4][5]. Таким образом, на практике может быть оправдано использование облачных функций для обслуживания чат-ботов и вебсервисов, от которых не требуется высокая производительность.

Иллюстрации

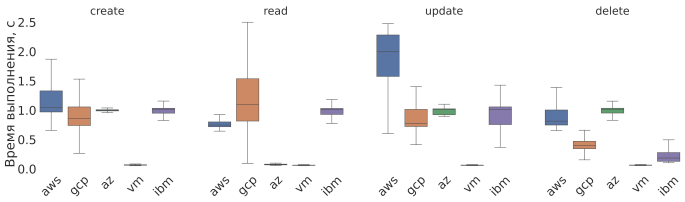

Рис.1 Время выполнения CRUD операций на различных платформах

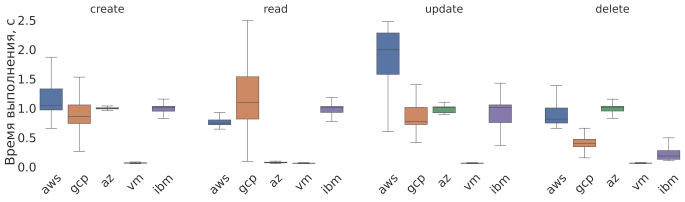

Рис.2 Время выполнения CRUD операций на различных платформах за вычетом круговой задержки

Литература

1. Savage N. Going serverless // In Communications of the ACM, 61(2), 2018, P. 15-16
2. Исходные коды обработчиков: `https://github.com/shibaeff/CRUDTest`
3. Gottlieb N., State of the Serverless Community Survey Results: `https://www.serverless.com/blog/state-of-serverless-community`
4. Калькулятор облачных функций Microsoft Azure: `https://azure.microsoft.com/ru-ru/pricing/calculator/?service=functions`
5. Калькулятор облачных функций IBM Cloud: `https://cloud.ibm.com/functions/learn/pricing`