ПОТЕНЦИАЛ ИСПОЛЬЗОВАНИЯ БЕССЕРВЕРНОЙ АРХИТЕКТУРЫ ДЛЯ МАСШТАБИРУЕМОЙ ОБРАБОТКИ ДАННЫХ

Щебетов А.А.; Дубовик Е.А.

Институт информационных технологий Белорусского государственного университета информатики и радиоэлектроники
г. Минск, Республика Беларусь

Савенко А.Г. – старший преподаватель, м.т.н.

Аннотация. В данной работе описаны преимущества и недостатки решения задачи облачной обработки данных с использованием бессерверной архитектуры. Для примера рассмотрен сервис AWS Lambda от облачного провайдера Amazon Web Services. Также в архитектуре для хранения входных и выходных данных были использованы такие сервисы, как: AWS Simple Storage Service, AWS Relational Database Service. Оценка релевантности применения данного подхода произведена по таким характеристикам, как: скорость разработки, максимальный объем обрабатываемых данных, скорость обработки одного запроса, стоимость.

На сегодняшний день в сфере веб-разработки набирает популярность использования бессерверной архитектуры. Данный подход применяется ввиду его преимуществ в масштабируемости, производительности, экономии и возможности обрабатывать большое количество запросов. Для этого не нужно управлять инфраструктурой, на которой работает сервис. А автоматическое масштабирование позволяет обслуживать тысячи одновременных запросов в секунду.

Бессерверная архитектура — это технологическое решение, основанное на событиях и запросах, которое позволяет разработчикам приложений создавать в облаке эффективные рабочие среды, обладающие всеми вычислительными ресурсами, необходимыми для организации бесперебойного процесса разработки. Подобные фреймворки очень удобны, особенно в условиях сжатых сроков и ресурсоемкости поставленных задач. Более того, выбор бессерверных сервисов позволяет оптимизировать процессы разработки приложений и таким образом повысить результативность других практик по оптимизации бизнес-процессов.

Бессерверная архитектура выглядит гораздо более перспективной для разработчиков приложений, поскольку она обеспечивает работу облачных рабочих сред по требованию. Это означает, что бессерверные функции запускаются только в момент фиксации определенного события. После этого функции выполняют последовательность операций в зависимости от команд, получаемых от пользователей. Затем бессерверная платформа применяет набор заранее подготовленных алгоритмов и правил, выполняет вычисления и выдает актуальные результаты [1].

AWS Lambda — это событийно-ориентированный сервис бессерверных вычислений, который позволяет выполнять код без выделения и администрирования серверов и дополнять другие сервисы AWS на основе пользовательской логики. Lambda автоматически реагирует на различные события (так называемые триггеры), например, на HTTP-запросы через Amazon API Gateway, изменение данных в корзинах Amazon S3 или таблицах Amazon DynamoDB; либо можно запустить свой код через вызовы API, используя AWS SDK и переходы между состояниями в AWS Step Functions [2].

Lambda выполняет код на высокодоступной вычислительной инфраструктуре и полностью отвечает за администрирование нижележащей платформы, включая обслуживание серверов и операционной системы, выделение ресурсов, автоматическое масштабирование, мониторинг кода и ведение журналов. То есть достаточно загрузить свой код и настроить каким образом и когда он должен выполняться. В свою очередь, сервис позаботится о его запуске и обеспечит высокую доступность вашего приложения.

Каждая функция работает в одной или нескольких выделенных средах, которые существуют лишь в течение жизненного цикла этой функции, а затем уничтожаются. В каждой среде одновременно выполняется лишь один вызов, но она используется повторно, если возникает множество серийных вызовов одной и той же функции. Все среды выполнения работают на виртуальных машинах с аппаратной виртуализацией — на так называемых виртуальных микромашинах (MicroVM). Каждая виртуальная машина назначается конкретной учетной записи AWS и может многократно использоваться средами для выполнения различных функций в этой учетной записи. Виртуальные машины упаковываются в структурные блоки аппаратной платформы Lambda Worker, которой владеет и управляет AWS. Одна и та же среда выполнения не может использоваться разными функциями, равно как виртуальные машины уникальны для разных учетных записей AWS. Модель функционирования AWS Lambada представлена на рисунке 1.

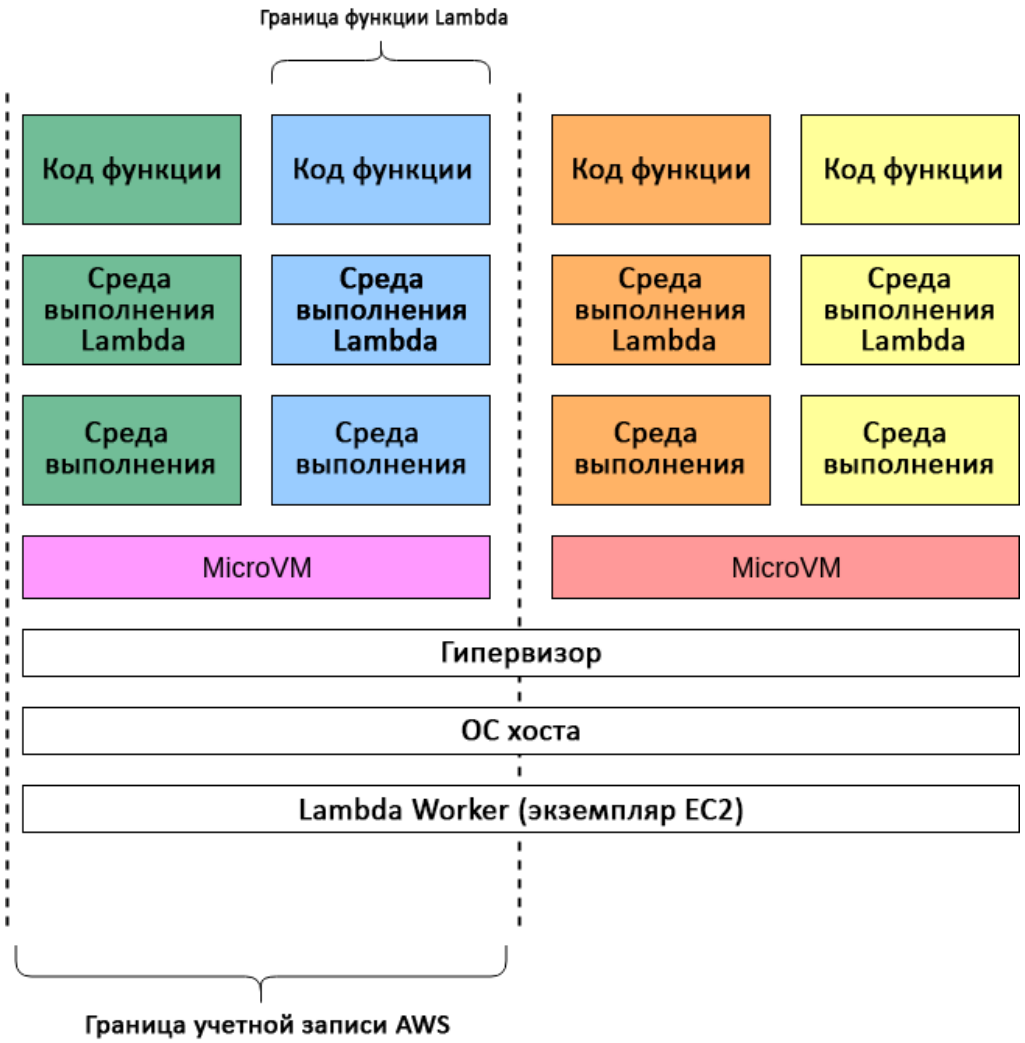

Рисунок 1 – *Модель изоляции в AWS Lambda*

Lambda идеально подходит для создания программных интерфейсов, а если использовать сервис вместе с API Gateway, можно значительно сократить расходы и быстрее запустить программное обеспечение в работу. Есть различные способы использования функций Lambda и варианты организации бессерверной архитектуры — каждый сможет выбрать что-то подходящее с учетом поставленной цели.

Придерживаясь бессерверной архитектуры можно создавать сервис-ориентированные действия, которые не выполняются постоянно. Типичный пример — масштабирование изображений.

У бессерверного подхода есть и свои минусы, такие как невозможность управления операционной системой на который выполняется код, отсутствие возможности контролировать процессор, память и ресурсы. Всем этим занимается провайдер. Также среди недостатков можно отметить такие, как: высокие требования к взаимодействию программных модулей и обратной совместимости, сложность мониторинга и отладки программы и привязанность к выбранному провайдеру [3].

Использование бессерверной архитектуры имеет большой потенциал и существенные преимущества по определенным характеристикам в сравнении с обычным подходом с одной серверной машиной. Поэтому решение о применении данного подхода должно опираться на конкретную задачу и быть обоснованным преимуществами для определенного случая.

Список использованных источников:

1. Ильченко В. Бессерверная архитектура или микросервисы — как выглядит будущее вычислительных технологий для бизнеса. / Habr [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/558990/ – Дата доступа: 05.04.2022.

2. Облачные вычисления с помощью AWS / Amazon Web Services, Inc. [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/lambda/ – Дата доступа: 05.04.2022.

3. Велинов Г. Детальный разбор AWS Lambda. / Habr [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/otus/blog/466519/ – Дата доступа: 05.04.2022.