УДК 004.75

ПРОЕКТИРОВАНИЕ ВЫСОКОНАГРУЖЕННОЙ РАСПРЕДЕЛЕННОЙ АРХИТЕКТУРЫ ДЛЯ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ В РАМКАХ IoT

А.В. Чуешев
Алтайский государственный технический университет им. И.И. Ползунова
г. Барнаул

В статье рассматривается проектирование и возможность использования облачной инфраструктуры как группы распределенных и виртуализированных сервисов для параллельной обработки и анализа большого объема данных в сфере IoT.

Ключевые слова: распределенная архитектура, облачные вычисления, микросервисная архитектура, событийно-ориентированная архитектура.

Начиная с 2010 года, с повсеместным распространением беспроводных сетей, появлением облачных вычислений, развитием технологий межмашинного взаимодействия и активным переходом на IPv6 происходит наполнение концепции «интернета вещей» (англ., Internet of Things, IoT) многообразным технологическим содержанием и практическими решениями (напр., Google Home, Apple HomeKit, Uber's self-driving car) [11].

Кроме того, необходимо отметить ежегодный рост в геометрической прогрессии количества производимой информации различных видов, в том числе в сфере IoT от всевозможных устройств, сенсоров и датчиков. Несмотря на появление большого количества публичных IaaS (AWS, Azure, GCP), предоставляющих различные инструменты для облачных вычислений, в ряде случаев согласно бизнес требованиям, принятых в компании, необходимо наличие локальной инфраструктуры для обработки значительного объема информации на распределенном кластере, построения различных математических моделей и с возможностью интеграции уже имеющих в компании сервисов.

Цель данной работы заключается в разработке архитектуры облачной платформы с возможностью автоматического развертывания как на публичных IaaS, так и на проприетарной инфраструктуре с предоставлением решений для распределенной аналитики и хранения данных, включая обработку информации в режиме реального времени (англ., stream processing).

Прежде чем перейти непосредственно к рассмотрение разработанной архитектуры необходимо проанализировать различные подходы, применяемые на этапе проектирования распределенных систем.

Отметим, что при условии разделения описываемых систем на набор распределенных, слабо связанных заменяемых компонентов (обычно web-службы), оснащенных стандартизированными интерфейсами для взаимодействия по регламентированным протоколам термин «распределенная архитектура» может быть сведен к более узкому понятию сервис-ориентированной архитектуры (англ., service-oriented architecture, SOA) [10].

В рамках SOA в качестве подмножества некоторые специалисты выделяют особую распределенную микросервисную модель, которая представлена набором небольших распределенных сервисов (независимые процессы), взаимодействующих с использованием легковесных механизмов коммуникации [6]. В качестве описанных механизмов могут быть использованы, например, следующие технологии:

– SOAP с WSDL;
– REST API;
– брокеры сообщений RabbitMQ, ActiveMQ с использованием различных протоколов для стандартизации сообщений (напр., STOMP) и т.д.

Для построение описываемой облачной платформы в качестве технологии, являющейся механизмом коммуникации между микросервисами, был выбран распределенный брокер сообщений Apache Kafka, вариант использования которого представлен на рисунке 1. Данный брокер был изначально разработан компанией LinkedIn под высоконагруженные проекты с поддержкой сотни тысяч сообщений в секунду, системы репли-

кации данных и гарантированной доставки сообщений со сложностью O(1) на дисковые структуры [3].

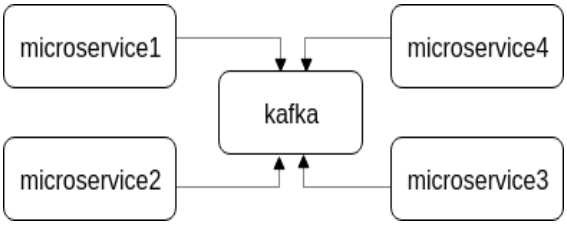

Рисунок 1 – Схема коммуникации микросервисов с использованием Apache Kafka

Комбинацию микросервисного подхода и брокера сообщений в качестве механизма коммуникации между отдельными элементами системы в специализированных источниках принято называть событийно-ориентированной архитектурой (англ., event-driven architecture, EDA) [2, 8]. Использование данного подхода в разработке распределенных систем позволяет достичь следующих результатов:

– изоляция команд разработчиков в соответствии с разрабатываемым микросервисом согласно правилу Amazon «two pizza»;

– частичное развертывание и обновление микросервисов в системе;

– возможность контейнеризации процессов в рамках Kubernetes;

– подключение к системе кластера Apache Spark, Data Lake или других сторонних систем;

– спецификация сообщений в системе на основе JSON схем и контента в рамках инструмента Apache Avro;

– нативная реализация паттернов CQRS, Event-Sourcing [4, 5].

Немаловажной частью описываемой платформы являются уровни безопасности и публичного доступа к микросервисам.

Одним из способов создания уровня безопасности является имплементация протокола OAuth2.0 в рамках отдельного сервиса с поддержкой Json Web Tokens (JWT). Данный метод предполагает передачу необходимой открытой информации (напр., идентификатор пользователя) в составе токена для проведения аутентификации в рамках отдельных микросервисов без совершения лишних запросов на уровень безопасности. Валидация токена на предмет подмены или изменения производится с использованием публичного RSA ключа, разосланного предварительно всем требуемым микросервисам со стороны уровня безопасности. Важно отметить, что, несмотря на процесс подписи токена при первом обращении на уровень безопасности закрытым RSA ключом и валидации открытым, информация, передаваемая в качестве контента, остается закодированной, но не зашифрованной. Более подробно данный процесс представлен на рисунке 2. Использование данного подхода позволяет строить взаимодействие между микросервисами с позиции недоверия (англ., non-trusted).

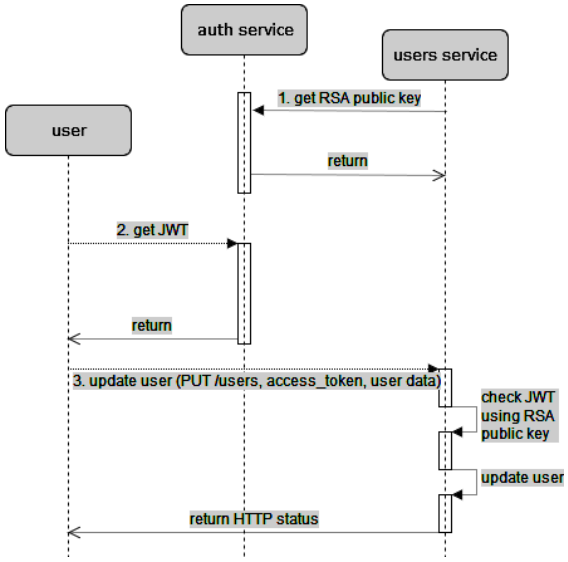

Рисунок 2 – Процесс аутентификации в системе

Публичный доступ к логике микросервиса может быть реализован следующими способами:

– индивидуальное API, реализованное в рамках каждого микросервиса;

– API Gateway, реализованный для группы микросервисов или системы в целом;

– реализация API для брокера сообщений Apache Kafka [7].

Выбор конкретного способа зависит от предъявляемых требований и уровня доверия, но необходимо отметить, что процесс получения публичного доступа происходит в тесном взаимодействии с уровнем безопасности, реализованном в качестве middleware.

Таким образом, было разработано 3 варианта архитектур облачной платформы для IoT, представленные на рисунках 3-5.

Первый вариант облачной платформы позволяет запускать кластеры микросервисов, реализованные посредством технологии Kubernetes, с поддержкой уровней безопасности (auth), регистрации и поиска микросервиса в системе (SD). Кроме того, подразуме-

вается балансировка нагрузки на уровне платформы за счет дублирования необходимых микросервисов и дальнейшей их регистрации.

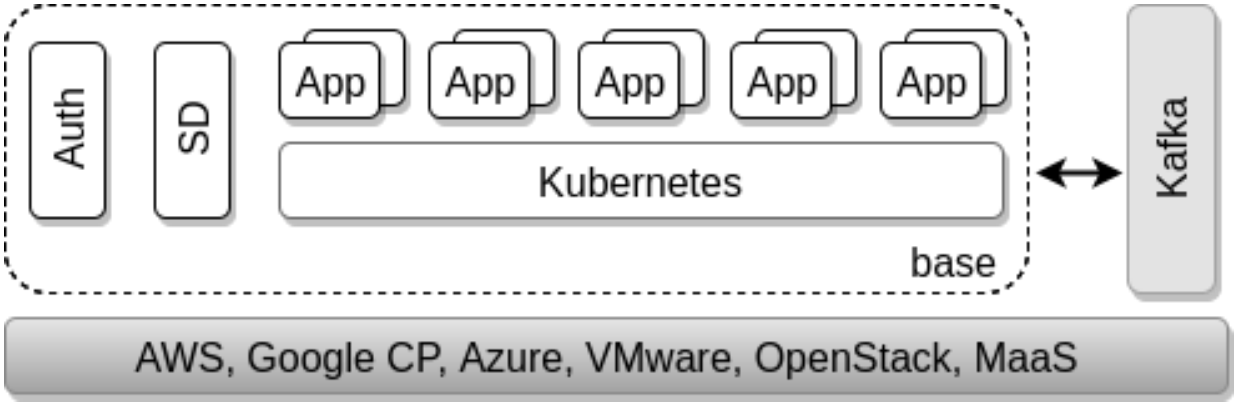

Рисунок 3 – Вариант №1 архитектуры облачной платформы

Второй вариант описываемой платформы дополнительно предоставляет реализацию Data Lake, основанную на Hadoop Distributed File System (HDFS) и Apache Cassandra, для хранения информации:

– неструктурированной;
– слабоструктурированной;
– полноценно структурированной.

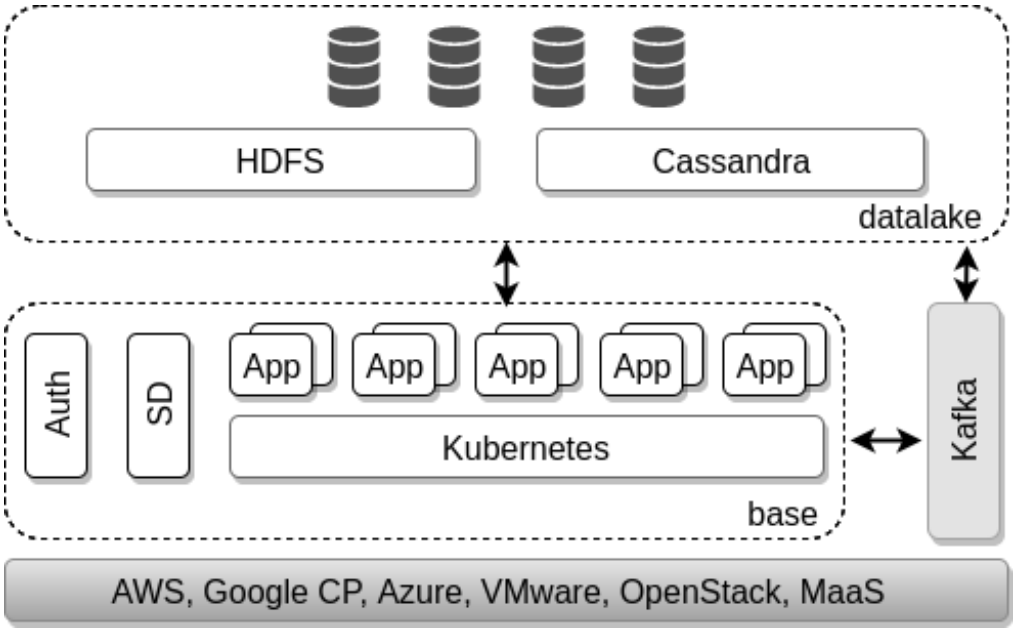

Рисунок 4 – Вариант №2 архитектуры облачной платформы

Использование третьего варианта облачной платформы позволяет дополнительно в комплексе с Data Lake задействовать кластер для распределенной аналитики, реализованного с использованием Apache Spark и подразумевающего использование:

– MLlib для распределенного машинного обучения;
– GraphX для обработки информации в виде графов;
– Spark Streaming для потоковой обработки данных;
– SparkSQL для эффективной работы с корпусом данных в терминах BigData [1, 9].

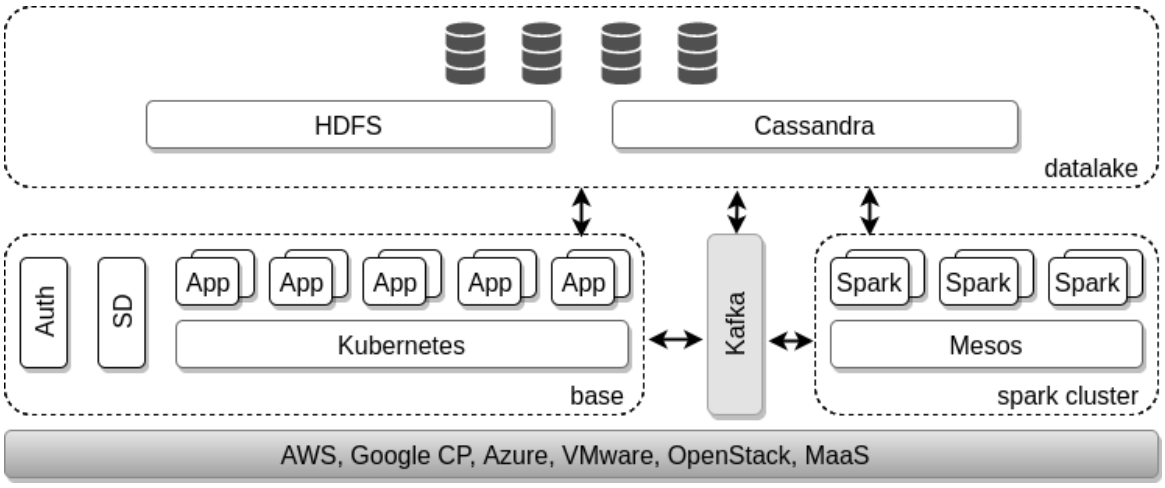

Рисунок 5 – Вариант №3 архитектуры облачной платформы

В заключение необходимо отметить, что развертывание разработанных вариантов архитектур облачной платформы осуществляется с использование средств оркестрации Juju от компании Canonical и соответствующих конфигураций в рамках концепции данного инструмента charms и bundles.

Итоговая пропускная способность платформы в единицу времени в значительной степени зависит от конфигурации Apache Kafka и физического железа.

СПИСОК ЛИТЕРАТУРЫ

1. Apache Spark [Электронный ресурс]. – Режим доступа : https://spark.apache.org/
2. David Luckham The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems / David Luckham. - Addison-Wesley Professional, 2002. – 400 p.
3. Kafka Ecosystem at LinkedIn [Электронный ресурс]. – Режим доступа : https://engineering.linkedin.com/blog/2016/04/kafka-ecosystem-at-linkedin
4. Kubernetes [Электронный ресурс]. – Режим доступа : https://kubernetes.io/docs/home/
5. Martin Fowler Event Sourcing [Электронный ресурс]. – Режим доступа: https://martinfowler.com/eaaDev/EventSourcing.html
6. Martin Fowler Microservices [Электронный ресурс]. – Режим доступа: https://martinfowler.com/articles/microservices.html
7. Pattern: API Gateway / Backend for Front-End [Электронный ресурс]. – Режим доступа: http://microservices.io/patterns/apigateway.html
8. Pattern: Event-driven architecture [Электронный ресурс]. – Режим доступа : http://microservices.io/patterns/data/event-driven-architecture.html
9. Sandy Ryza Advanced Analytics with Spark: Patterns for Learning from Data at Scale / Sandy Ryza, Uri Laserson, Sean Owen, Josh Wills. - O'Reilly Media, 2015. – 276 p.
10. Understanding Service-Oriented Architecture [Электронный ресурс]. – Режим доступа: https://msdn.microsoft.com/en-us/library/aa480021.aspx
11. Википедия. Свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Internet_of_things.

Чуешев Александр Витальевич – магистрант, тел.: +79627982437, e-mail: alexchueshev@gmail.com.