BITWORKS
SOFTWARE COMPANY

Анализ и агрегация сетевого трафика с целью обнаружения и прогнозирования аномалий при помощи Apache Kafka, Apache Spark и InfluxDB

/Сентябрь 2017/

СОДЕРЖАНИЕ

Компания Bitworks занимается проектированием и разработкой комплексных программных систем. Для задач в сложных предметных областях мы часто разрабатываем наиболее подходящую архитектуру путем создания действующего прототипа с использованием проверенных и надежных открытых технологий. Итоговое решение максимально соответствует индустриальным стандартам и отвечает требованиям надёжности, производительности и безопасности.

ЗАДАЧА

Описание проблемы

Один из наших клиентов занимается обслуживанием большой компьютерной сети. Для обеспечения работоспособности инфраструктуры необходимо осуществлять наблюдение текущих и прогнозируемых тенденций сетевой активности, анализировать потоки данных и предсказывать возможные проблемы.

Однако, даже при использовании стандартных статистических методов sFlow, информация, содержащаяся в первичном потоке данных, трудно идентифицируется и не может быть использована для анализа без предварительной обработки.

Перед нами была поставлена задача разработать масштабируемую систему для агрегации, обогащения и анализа больших потоков сетевых данных.

Закономерности, обнаруженные в потоках данных, могут стать основанием для будущей реструктуризации сети и модернизации оборудования, в то время как аномальные изменения в структуре трафика могут означать аппаратный сбой или злонамеренное поведение и требовать оперативных мер

Свойства потока данных sFlow

При проектировании прототипа мы исходили из следующих предположений о характере данных:

- Поток данных является непрерывным.
- Данные являются временным рядом. Каждый элемент данных имеет временную отметку, и элементы поступают в систему в порядке, соответствующем порядку отметок.
- Данные являются необработанными. Часть полей могут быть исключены из рассмотрения в зависимости от задач анализа, в то время как другие поля должны быть дополнены данными из внешних источников.
- Данные содержат множество полей, в том числе полей-признаков (которые позволяют классифицировать и секционировать данные) и полей-значений (которые количественно характеризуют элементы данных).
- Допустима повторная обработка данных. Исходные данные и целевые метрики носят статистический характер, поэтому происходящая время от времени (в случае сбоя) повторная обработка одних и тех же значений не влияет на конечный результат.

Требования к обработке данных

Разрабатываемая система должна обрабатывать первичные данные в пределах временного окна следующим образом:

- Отбрасывать ненужные поля.
- Именовать поля по необходимости.
- Создавать новые поля путем преобразования исходных полей и применения базовых операций (сложение, вычитание, умножение, деление).
- Агрегировать поля-значения по ключевым признакам при помощи одной из агрегирующих операций (max/min, сумма, количество, количество уникальных и т.д.).

Полученные в результате данные обычно более компактны, однако, они по прежнему являются временными и имеют много измерений, поэтому для хранения и обеспечения оперативного доступа к агрегированным данным необходимо специализированное постоянное хранилище.

Постоянное хранилище данных должно быть оптимизировано для следующих операций:

- Добавление записей за ближайший промежуток времени (порядок добавления может не соответствовать порядку временных отметок).
- Чтение упорядоченных записей за непрерывный период времени с возможностью фильтрации по одному или нескольким полям-признакам.
- Удаление данных за непрерывный период времени по истечении срока их актуальности.

Обработанные данные необходимо анализировать, для того чтобы:

- Обнаруживать аномалии и закономерности и уведомлять о них;
- Предсказывать тенденции.

Анализ должен производиться при поступлении новых данных с использованием исторических данных из постоянного хранилища. Методы анализа заранее не определены, поэтому в системе должна быть реализована поддержка пользовательских функций анализа.

Помимо этого необходим инструмент визуального анализа накопленных данных с возможностью построения графиков в реальном времени.

Результирующая система должна быть масштабируемой и легко развертываемой.

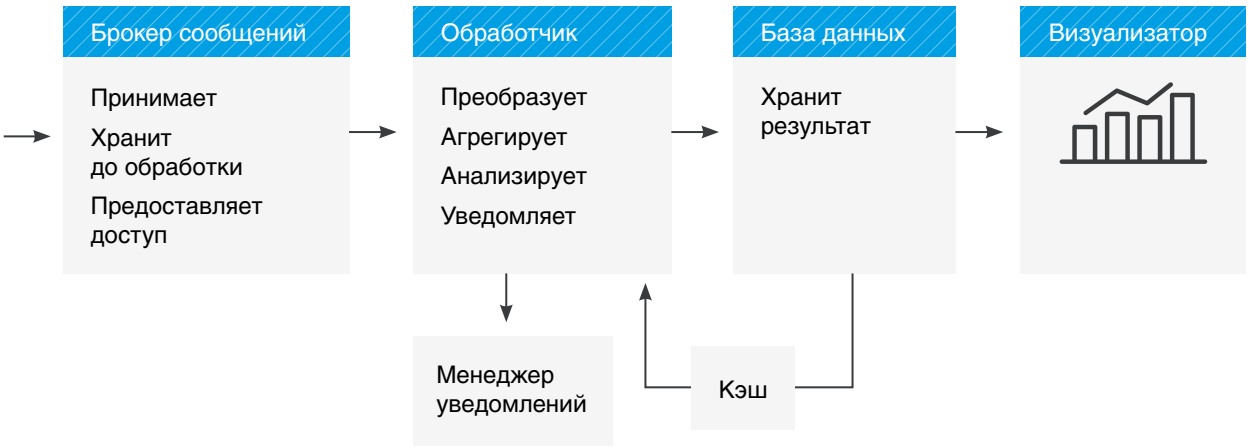
РЕШЕНИЕ

Высокоуровневая архитектура

В соответствии с указанными требованиями система должна состоять из следующих подсистем:

- Подсистема потоковой передачи для сбора, упорядочивания и секционирования поступающих данных.
- Подсистема потоковой обработки для трансформации и анализа данных.
- База данных временных рядов для хранения результирующих данных и предоставления доступа к ним для последующего анализа.
- Инструмент визуального анализа.
- Система доставки уведомлений.

Была предложена следующая высокоуровневая архитектура.

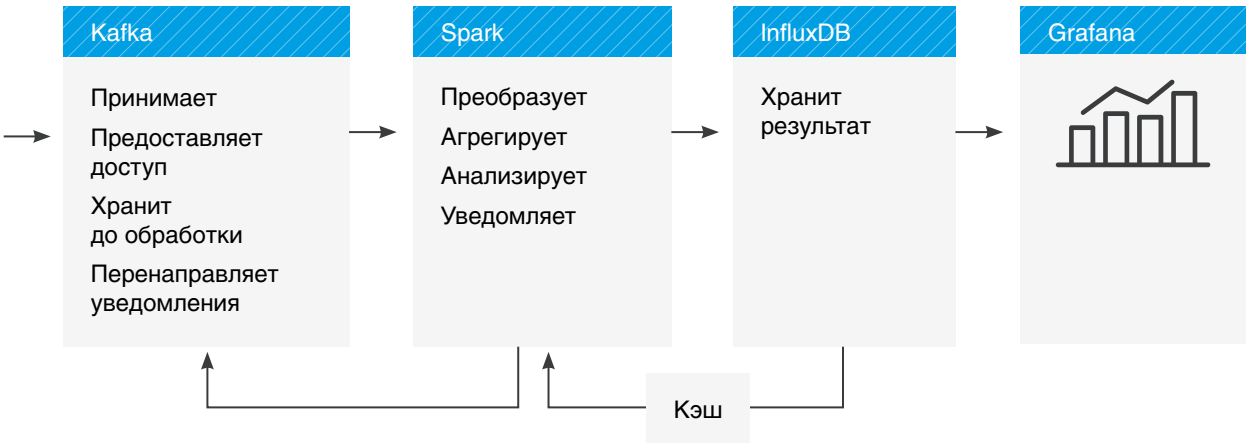


Архитектура действующего прототипа

Было принято решение разработать действующий прототип для определения соответствия этой архитектуры требованиям клиента. В качестве конкретных компонентов мы выбрали:

- **Python 3 в качестве языка разработки.** Python 3 является открытым языком программирования, который активно поддерживается научным сообществом. Python 3 является одним из немногих языков, поддерживаемых Apache Spark. Этот выбор был продиктован клиентом.
- **Kafka для передачи исходных потоков данных.** Apache Kafka является открытой распределенной потоковой системой, часто используемой в тандеме со Spark. Она объединяет потоки событий из множества источников, обеспечивает их сохранность даже в случае сбоя и гарантирует доставку сообщений «хотя бы один раз». Kafka поддерживает множество одновременных потребителей данных, а также позволяет гибко настраивать масштабируемое секционирование данных относительно структуры потенциальных источников и потребителей.
- **Spark для обработки данных.** Apache Spark является открытой распределенной системой для обработки данных, которая стала де-факто стандартом для решения задач в области больших данных. Spark предоставляет унифицированную модель обработки массивов и потоков данных. Её API доступно для языка программирования Python и содержит методы для трансформации и агрегации данных, а также большую библиотеку алгоритмов машинного обучения для анализа данных.
- **InfluxDB для хранения данных.** InfluxDB является открытой распределенной СУБД для хранения временных рядов. Она оптимизирована для хранения и обеспечения доступа к многомерным временным данным, обходит большинство своих конкурентов в производительности и является самой популярной среди СУБД обработки временных рядов согласно DB-Engines Ranking.
- **Grafana для визуального анализа данных.** Grafana является открытой системой для мониторинга и анализа, которая предоставляет набор инструментов для визуализации временных данных, а также имеет обширную базу плагинов сторонних производителей. Grafana имеет встроенную поддержку базы данных InfluxDB (с интерактивным построителем запросов и другими функциями), а также других источников данных.

- **Docker для развертывания приложения.**
  Docker является открытой системой для сборки, поставки и запуска распределенных приложений. Docker позволяет гибкую конфигурацию систем с множеством компонент, а также гарантирует корректное выполнение независимо от целевой платформы развертывания. InfluxDB и Grafana имеют официальные образы Docker, а для Kafka существует образ, поддерживаемый сообществом.



Наша компания обладает существенным опытом работы с выбранными компонентами в различных комбинациях, поэтому было решено проверить их состоятельность в рамках представленной архитектуры.

Компонент	Лицензия
Apache Kafka – брокер сообщений	Apache License 2.0
Apache Spark – платформа для распределенной обработки	Apache License 2.0
InfluxDB – база данных временных рядов	MIT License for manual deployment ToS for SaaS Software License Subscription Agreement for SaaS and/or scalability
Grafana – платформа для наблюдения и анализа	Apache License 2.0 for manual deployment ToS for SaaS
Docker – платформа для автоматизации развертывания	Apache License 2.0
Python – язык программирования	License

Реализация

Было разработано приложение на языке Python, которое запускается в кластере Spark.

В первую очередь приложение подключается к серверу Kafka в качестве потребителя, используя настройки, указанные в конфигурационном файле.

Затем приложение обрабатывает каждый пакет данных, полученный из Kafka, в два этапа:

1. Преобразование и дополнение данных

   - Выбор полей данных, необходимых для текущей задачи.
   - Переименование полей по необходимости.
   - Базовые арифметические вычисления (сложение, вычитание, умножение, деление).
   - GeoIP трансформация (страна, город, ASN).

   Вложенные трансформации и пользовательские функции не поддерживаются в текущей версии прототипа.

2. Агрегация данных с использованием поддерживаемых агрегирующих функций. В рамках прототипа были реализованы следующие функции:

   - Sum, возвращает сумму всех значений поля в пакете данных.
   - Mult, возвращает произведение всех значений.
   - Max, возвращает наибольшее значение.
   - Min, возвращает наименьшее значение.

   Агрегация возможна по ключевому полю (или комбинации полей), или для всего пакета данных целиком.

Способ преобразования данных (порядок трансформации и вид агрегации) не является заранее определенным и может быть описан в конфигурационном файле при помощи DSL.

Данные, полученные в результате обработки, отправляются на хранение в InfluxDB.

Приложение анализирует обработанные данные с использованием встроенных и пользовательских модулей анализа. Эти модули запрашивают накопленные данные из InfluxDB, анализируют их на предмет наличия отклонений и отправляют уведомление в случае обнаружения аномалии.

Пользовательский модуль должен быть написан на Python 3, представлять из себя пакет с классом, реализующим определенный интерфейс. После этого можно указать этот модуль и параметры его инициализации в конфигурационном файле. Каждый модуль может быть использован несколько раз с различными параметрами.

Пользовательский модуль должен быть написан на Python 3, представлять из себя пакет с классом, реализующим определенный интерфейс. После этого можно указать этот модуль и параметры его инициализации в конфигурационном файле. Каждый модуль может быть использован несколько раз с различными параметрами.

Текущий прототип позволяет отправлять уведомления в отдельный сервер и/или топик Kafka.

GitHub

Более подробное описание архитектуры прототипа доступно в GitHub.

РЕЗУЛЬТАТ

В рамках предложенной высокоуровневой архитектуры и с использованием выбранных компонентов мы разработали действующий прототип, который доказал применимость подхода для количественного и статистического анализа сетевого трафика в формате sFlow.

Прототип опубликован на GitHub под лицензией Apache License v2 и может быть использован без ограничений в других проектах. Он может быть использован без изменений для анализа любых потоков данных в формате CSV в рамках указанных выше операций.

Возможные сферы применения

Аналогичный подход может быть использован и для других задач потоковой агрегации и анализа. Список возможных сфер применения включает в себя, помимо прочего:

Телекоммуникация

- Мониторинг сетевой нагрузки
- Обнаружение и прогнозирование DDoS атак

Devops

- Мониторинг эксплуатации аппаратных средств
- Мониторинг логов

Финансы

- Анализ рынка ценных бумаг
- Управление пакетами ценных бумаг

Транспорт

- Управление транспортными потоками
- Служба заказа такси

Интернет вещей

- Климат-контроль
- Умный город
- Технологические операции

Разработка программного обеспечения

- Статистика использования приложений
- Обнаружение вредоносного программного обеспечения

Целесообразность представленной архитектуры для конкретных задач в упомянутых сферах необходимо проверять отдельно для каждого случая. Мы будем признательны получить обратную связь о вашем опыте использования рассмотренного подхода. Далее представлены некоторые рекомендации, которые помогут определить, подойдет ли предложенное решение для некоторой конкретной задачи.

Рекомендации по применению

Предложенная архитектура скорее всего подойдет для задач обработки больших данных, в которых:

- Существует множество автоматизированных событий во времени (показания счётчиков, пакеты данных, сообщения и т.д), которые необходимо контролировать и анализировать.
- Данные носят временной характер и содержат известный набор полей с определенными доменами значений.
- Данные необходимо преобразовать перед обработкой и хранением.
- Необходимо обнаруживать нестандартное поведение, выявлять тенденции и прогнозировать будущие состояния данных.
- Данные необходимо агрегировать, для того чтобы избежать масштабирования системы хранения.
- Apache Spark уже является частью портфеля технологий компании.
- Вычисления нечувствительны к возможным дублирующим значениям.

Предложенная архитектура не подойдет для задач, в которых:

- Необходима обработка в реальном времени с минимальной задержкой. Использование фреймворка Spark Streaming для потоков данных возможно только при пакетировании событий, поэтому время отклика не может быть меньше чем окно пакетирования.
- Задача требует точного численного решения. В предложенном решении в случае сбоев обработки данные могут быть обработаны более одного раза, что приведет к некорректному результату. Kafka 0.11 предоставляет возможность обработки данных «только один раз» для своих производителей и потребителей. Помимо этого приложение должно обеспечить идемпотентность добавления преобразованных данных в постоянное хранилище.
- Данные не являются большими. В этом случае нет необходимости в масштабируемости, предоставляемой архитектурой, и должно подойти более простое решение.
- Нет необходимости в сложном анализе. InfluxDB предоставляет возможности для трансформирования и агрегации данных, в то время как Kapacitor может быть использован для простых проверок на аномалии. При этом можно использовать также и Grafana для текущего контроля данных.

Для обеспечения идемпотентности операций над данными необходимы дополнительные меры при получении и при сохранении данных. Kafka 0.11 предоставляет возможность обработки данных “только один раз” для своих производителей и потребителей. Помимо этого приложение должно обеспечить идемпотентность добавления преобразованных данных в постоянное хранилище.

Связка InfluxDB и Grafana может лучше подойти для такой задачи.

СОСТАВИТЕЛИ ДОКУМЕНТА

Сергей Крицкий

Руководитель команды разработки

Денис Рябоконь

Разработчик

Николай Богословский

Разработчик

Вячеслав Подбережный

Проектный менеджер

Валентин Рахимов

Системный аналитик

13A Shishkova St,
Tomsk, 634050, Russia

+7 (382) 2 70 54 77

30 Regent St,
Jersey City, NJ 07302, USA

+1 929 402 8251

Hamburg, Germany

+49 (040) 5 48 91 029

info@bw-sw.com

https://bitworks.software