Разработка отказоустойчивого конвейера для доставки данных в режиме квазиреального времени

Сафин А.Ф., СПбГУ, Санкт-Петербург st090196@student.spbu.ru,
Благов М.В., СПбГУ, Санкт-Петербург mikhail.blagov@gmail.com

Аннотация

В статье рассматривается разработка и тестирование отказоустойчивого потокового конвейера данных. В работе описан процесс проектирования и проверки надёжности системы. Особое внимание уделяется рассмотрению возможных сбоев и методам обеспечения отказоустойчивости, таким как использование отказоустойчивых компонентов и автоматическое восстановление после сбоев.

Введение

При решении задачи анализа данных возникает потребность в переносе данных с продуктивной базы данных в аналитическое хранилище, которое было создано специально для того, чтобы потреблять и структурировать данные для последующего анализа [1].

Аналитическое хранилище облегчает анализ данных, не создавая нагрузки на источники, которые обеспечивают повседневную работу. Обычно, данные в аналитическом хранилище обновляются периодическим образом, например, еженедельно или ежедневно. Загрузка данных в аналитическое хранилище, как правило, запланирована на непиковые часы, когда как источники, так и хранилище испытывают низкую нагрузку, например, в ночное время. Таким образом, оно хранит данные по состоянию на прошлую неделю или на вчерашний день, в то время как текущие данные доступны только в источниках [2].

Однако бизнес-пользователи часто нуждаются в актуальных данных для поддержки своевременного принятия решений. Данная задача решается при помощи потоковых систем обработки данных, которые позволяют в режиме квазиреального времени переносить данные из источника в аналитическое хранилище [2]. На практике поток данных часто не ограничен, информация приходит постепенно, с течением времени [3]. Важно обработать каждое сообщение в отдельности таким образом, чтобы оно хотя бы один раз [4] дошло из источника в хранилище.

Также необходимо, чтобы доставка данных происходила с минимальными задержками и максимальной надежностью. Данная работа посвящена тому, чтобы разработать отказоустойчивый потоковый конвейер обработки данных, используя популярные инструменты, и показать его устойчивость к сбоям.

Проектирование конвейера данных

Целью работы является построение и реализация принципов отказоустойчивости конвейера для доставки данных в режиме квазиреального времени, которые позволят потоковому приложению продолжать функционировать надежно и эффективно даже при возникновении сбоев. Это включает в себя проектирование системы таким образом, чтобы она могла обнаруживать и восстанавливаться от сбоев.

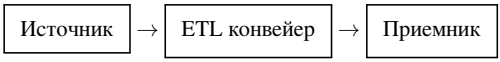

Рис. 1: Верхнеуровневая схема конвейера данных

Рассмотрим конвейер данных, изображенный на рис. 1. Такой конвейер данных является распределенной системой. Рассмотрим возможные типы неисправностей в распределенных системах.

Типы неисправностей в распределенных системах

Распределенные системы характеризуются высокой сложностью из-за большого количества компонентов и использованием различных аппаратных и программных платформ [5]. Эта сложность делает их склонными ко многим проблемам. Выделяют три типа проблем, которые могут возникнуть в распределенной системе:

- Сбой – это неожиданное или ненормальное поведение компонента системы, которое может привести к ошибке или сбою [5].
- Ошибка – ошибочное состояние системы, возникающее в результате сбоев [5].
- Отказ определяет событие, при котором система не может предоставить услугу или выполнить намеченную цель. Это видимый результат ошибки [5].

Сбой в одном узле может распространиться по всей системе. Поэтому важно обеспечить, чтобы она продолжала работать даже при наличии неисправностей. Это достигается путем предварительного анализа типов и учетом этих неисправностей при разработке системы [5].

Неисправности классифицируются по частоте их появления на временные, периодические и постоянные неисправности. Временные неисправности возникают один раз и исчезают, а периодические неисправности появляются и исчезают неоднократно. Что касается постоянных неисправностей, то они появляются и остаются до тех пор, пока не будут устранены [5].

Определение: Отказоустойчивость – это способность конвейера данных как распределенной системы функционировать должным образом даже при наличии временных и периодических сбоев [5].

Для конвейера, изображенного на рис. 1, реализуем принципы отказоустойчивости. Для него характерны следующие типы отказов:

- сбой источника данных;
- сбой сети между источником и ETL;
- сбой отдельных компонентов, виртуальных машин и контейнеров, на которых запущен ETL;
- сбой сети между ETL и приемником;
- сбой приёмника.

Будем рассматривать временные и периодические сбои, при которых функционирование компонента восстанавливается спустя некоторое время после первичного сбоя.

Архитектура конвейера обработки данных

На рис. 2 представлена схема конвейера. В ней можно выделить следующие элементы:

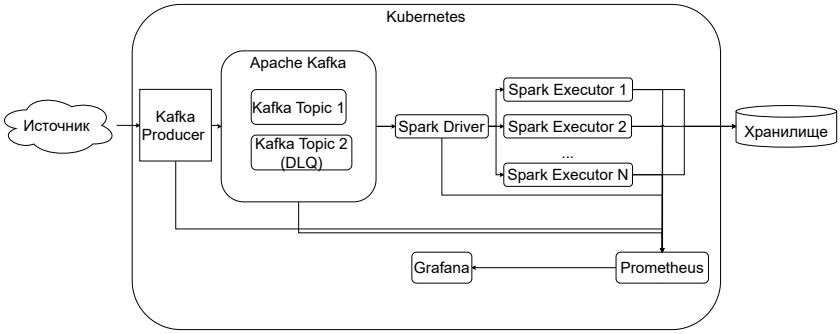

Рис. 2: Архитектура

- Источник – система, из которой конвейер читает данные;
- Набор блоков для обработки данных, развёрнутых в Kubernetes [6]:
  - Kafka Producer читает данные из источника и отправляет сообщения в Kafka.
  - Блок Kafka [7] отвечает за хранение сообщений, которые предстоит обработать: сообщения из Kafka Topic [8] 1 отправляются в Spark приложение, Kafka Topic 2 (DLQ) содержит сообщения, которые были некорректно обработаны и требуют повторной обработки для обеспечения “хотя бы один раз” семантики доставки данных.
  - Spark Driver [9] отвечает за три вещи: сохранение информации о приложении Spark; реагирование на программу пользователя или вводимые данные; и анализ, распределение и планирование работы между исполнителями.
  - Spark executors [9] несут ответственность за фактическое выполнение работы, которую им поручает Spark Driver. Это означает, что каждый Spark executor отвечает только за две вещи: выполнение кода, назначенного ему Spark Driver’ом, и отправку отчета о состоянии вычислений на этом исполнителе обратно на узел Spark Driver’a.
  - Блок мониторинга отвечает за наблюдаемость для конвейера, а также служит для оповещениях о сбоях. Данный блок реализован при помощи инструментов Prometheus [10], Grafana [11].
  - Система-приемник, в которую конвейер должен доставить данные.

Методика тестирования

Для тестирования отказоустойчивости применим подход, известный как Хаос-тестирование [12, 13]. Приложение будет работать какое-то время при ”идеальных” условиях. Далее, искусственно будем вносить в систему сбои и отслеживать, как контейнер реагирует на это.

- При сбое источника данных производится вывод предупреждения и повторная попытка соединения. После восстановления источника конвейер переподключается к нему автоматически.

- При отсутствии связи с источником Kafka Broker [7] продолжает функционировать, но не обрабатывает данные, поэтому необходимо просигнализировать о падении входного потока сообщений. После восстановления соединения конвейер продолжает работу.

- При сбое Kubernetes необходим перезапуск кластера вручную. При этом связь между подами [6] может потеряться. Использование StatefulSets [6] вместо стандартных Deployments [6] исключает обозначенную проблему.

- При сбое Kafka Profucer, под автоматически перезапускается. При этом возможна потеря сообщений. Использование механизма повторной отправки (retry mechanisms) [7] исключает обозначенную проблему.

- При сбое Spark Driver, под автоматически перезапускается, при этом нужно не потерять данные. Для этого необходимо дать Spark инструкцию фиксировать offsets [8] только после того, как произведена операция записи в хранилище. Необработанные же сообщения (которые не были записаны в хранилище) отправляются в резервный топик Kafka для повторной обработки.

- При сбое одного из Spark Executor он автоматически перезапускается, необработанные сообщения отправляются в резервный топик Kafka для повторной обработки. При этом оставшийся Executor автоматически продолжает обработку сообщений.

- При отсутствии связи с приемником, ETL конвейер продолжает функционировать, но сообщения отправляются в DLQ топик как необработанные. Поэтому необходимо просигнализировать о падении выходного потока сообщений. После восстановления соединения конвейер продолжает работу.

- При сбое приемника производится вывод предупреждения и повторная попытка соединения. После восстановления приемника конвейер переподключается к нему автоматически.

Заключение

В работе сформулированы принципы отказоустойчивости конвейера для доставки данных в режиме квазиреального времени. Рассмотрены основные типы отказов для таких конвейеров, реализовано потоковое приложение, проведено тестирование путем внесения временных сбоев в систему и показана отказоустойчивость разработанного конвейера в рассмотренных случаях.

Список литературы

[1] Что такое потоковая передача данных? // https://aws.amazon.com/ru/what-is/streaming-data/

[2] Jörg, T., Dessloch, S. Near Real-Time Data Warehousing Using State- of-the-Art ETL Tools // Lecture Notes in Business Information Processing. – Springer, 2009. – С. 100–117.

[3] Клеппман, М. Высоконагруженные приложения. Программирование, масштабирование, поддержка. – Санкт-Петербург: Питер, 2018. – 640 с.

[4] Bhimani, P., Panchal, G. Message Delivery Guarantee and Status Update of Clients Based on IoT-AMQP // Lecture Notes in Networks and Systems – 2017. – С. 15-22.

[5] Rhim, H. Fault and Failure in Distributed Systems // https://www.baeldung.com/cs/distributed-systems-fault-failure.

[6] Hightower, K., Burns, B., Beda, J. Kubernetes: Up and Running. 2nd edition. – Sebastopol, CA: O'Reilly Media, Inc., 2019. – 277 с.

[7] Kreps, J. Putting Apache Kafka to Use: A Practical Guide to Building a Stream Data Platform (Part 2). February 25, 2015. https://www.confluent.io/blog/stream-data-platform-2/

[8] Скотт, Д., Гамов, В., Клейн, Д. Kafka в действии. 1-е издание. – Москва: ДМК Пресс, 2022. – 310 с.

[9] Optimizing Spark performance on Kubernetes // https://aws.amazon.com/ru/blogs/containers/optimizing-spark- performance-on-kubernetes/

[10] What is Prometheus? // https://prometheus.io/docs/introduction/overview/

[11] What is Grafana? // https://www.redhat.com/en/topics/data- services/what-is-grafana

[12] Principles of Chaos Engineering // https://principlesofchaos.org/

[13] Lewis, J., Wang, C. Chaos Engineering: New Approaches To Security // Rain Capital, 2019.

[14] White, T. Hadoop: The Definitive Guide STORAGE AND ANALYSIS AT INTERNET SCALE. 4th edition. – Sebastopol, CA: O'Reilly Media, Inc., 2015. – 728 c.