Особенности организации конвейера данных с использованием брокера сообщений Apache Kafka в системах обработки данных

Селезнёв Александр Игоревич, ассистент;
Селезнёв Игорь Львович, кандидат технических наук, доцент
Белорусский государственный университет информатики и радиоэлектроники (г. Минск, Беларусь)

В статье рассматриваются особенности организации конвейера данных с использованием Apache Kafka в системах обработки данных. Обсуждаются основные структурные уровни конвейера данных, особенности построения, а также анализируется интеграция с Apache Kafka.

Ключевые слова: *система обработки данных, конвейер данных, брокер сообщений Apache Kafka.*

В настоящее время многообразие различных типов данных, используемых современными системами обработки данных (СОД), взрывообразно увеличилось. Этому явлению способствует все большее внедрение интеллектуальных устройств в промышленном и потребительском секторе с Интернетом вещей (Internet of things, IoT) в качестве связующего звена [1]. Следовательно, многократно возрастает поток информации, генерируемый большим числом

устройств, с разными форматами выходных данных, для чего интенсивно применяются конвейеры данных.

Конвейер данных — набор процессов и методов, используемых для перемещения данных из разных исходных систем в централизованное хранилище (обычно информационное), для анализа, обработки и дальнейшего применения [2]. Пример структуры типичного конвейера данных приведен на рисунке 1.

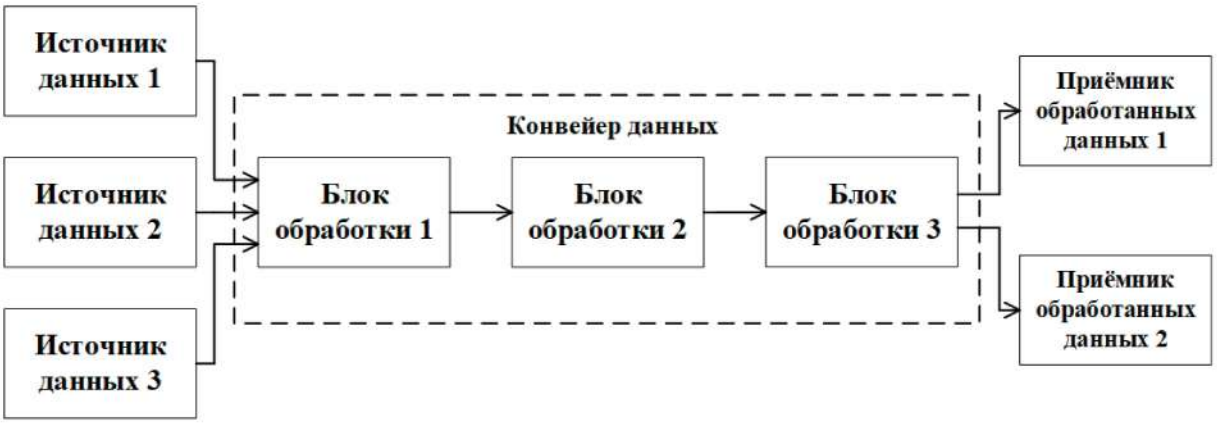

Рис. 1. **Пример структуры типичного конвейера данных**

Из рисунка 1 видно, что в конвейер поступают данные из трех источников, которые затем последовательно обрабатываются в блоках обработки конвейера. Данные из «Блок обработки 3» отправляются в два приёмника обработанных данных для их дальнейшего использования согласно настроенной внутренней логики конвейера данных. Блоки обработки данных могут быть реализованы как отдельными функциями обработки, так и полноценными системами обработки данных [3].

Структурно конвейеры данных состоят из пяти элементов, представленных на рисунке 2.

Рис. 2. **Структурные уровни конвейера данных**

На рисунке 2 приведены следующие структурные уровни конвейера данных:

1. Источники входных данных. Данные генерируются из различных источников, таких как, например, взаимодействие клиентов на веб-сайте, транзакции в розничном магазине, устройства IoT или любых других источников генерации данных.

2. Уровень приема данных. На этом уровне устанавливается соединение с источниками входных данных через соответствующие протоколы и соединители. После подключения необходимые данные извлекаются из каждого источника. Метод извлечения зависит от формата источника данных — структурированные данные можно получить с помощью запросов, а для неструктурированных данных чаще всего требуются специальные инструменты извлечения.

3. Уровень хранения данных. Принятые данные находятся в необработанной форме и должны быть сохранены перед дальнейшей их обработкой.

4. Уровень обработки данных включает процессы и инструменты для преобразования необработанных данных.

5. Уровень доставки данных и аналитики. Преобразованные данные загружаются в хранилище данных или другой репозиторий и становятся доступными для формирования отчетов и анализа.

Для организации производительного конвейера данных в СОД необходимо обеспечить надежность и безотказность уровня приёма данных посредством оптимального построения интерфейса взаимодействия между элементами конвейера. Решением этой задачи является как написание собственного интерфейса взаимодействия, так и использование существующих специализированных средств — брокеров сообщений [4]. Одним из оптимальных вариантов брокера сообщений, поддерживающим работу с конвейерами данных, является брокер сообщений Apache Kafka (далее Kafka).

Существует несколько сценариев использования Kafka: в конвейере данных этот брокер сообщений может представлять одну из конечных точек конвейера (например, перемещение данных из базы данных MongoDB в Kafka); также возможно создание конвейера данных между двумя

различными системами используя Kafka в качестве промежуточного звена (например, перемещение данных из социальной сети X в Elasticsearch путем отправки их сначала в Kafka, а затем из Kafka в Elasticsearch [5]).

Главным достоинством использования Kafka в конвейерах данных является то, что этот брокер сообщений может служить большим и надежным буфером между различными этапами конвейера. Тем самым Kafka разделяет производителей данных от потребителей внутри конвейера и позволяет использовать одни и те же данные из источника в нескольких целевых приложениях и системах, которые имеют различные требования к своевременности и доступности; обеспечивает высокую надежность и производительность — вследствие чего Kafka хорошо подходит для большинства конвейеров данных [6].

Основные особенности в организации производительных конвейеров данных с использованием Kafka в СОД:

1. Своевременность (Timeliness). Например, в одних системах ожидается, что данные будут поступать большими порциями раз в день, в других — через несколько миллисекунд после генерации. Хорошие системы интеграции данных должны соответствовать требованиям к своевременности для разных конвейеров, а также иметь возможность перехода от одного режима работы к другому при необходимости.

Kafka может использоваться для поддержки как конвейеров, работающих в режиме реального времени, так и для обработки пакетов, поступающих с меньшей интенсивностью. Производители могут записывать данные в Kafka с требуемой им частотой, а потребители могут читать и доставлять самые свежие события по мере их поступления.

2. Надежность. При создании конвейеров важно обеспечить быстрое автоматическое восстановление после сбоев и иметь как можно меньше критических точек отказа. Данные часто поступают по конвейерам в важные для компаний элементы работающей системы, и сбой длительностью более нескольких секунд может иметь катастрофические последствия, особенно если в требованиях к функционированию указаны величины порядка нескольких миллисекунд. Еще один важный фактор надежности — гарантия доставки данных. Хотя в некоторых системах потери данных допустимы, чаще всего требуется как минимум однократная их доставка. Это означает, что все данные, отправленные из системы-источника, должны достичь пункта назначения, что в отдельных случаях может привести к появление дубликатов из-за повторной отправки. Часто выдвигается требование строго однократной доставки — все данные, отправленные из системы-источника, должны достичь пункта назначения без каких-либо потерь или дублирования.

Kafka способен обеспечить как минимум однократную доставку, а в сочетании с внешним хранилищем с поддержкой транзакционной модели или уникальных ключей также и строго однократную [7]. Поскольку многие из конечных точек брокера сообщений Kafka представляют собой хранилища данных, обеспечивающие возможность строго однократной доставки, конвейер на основе Kafka можно сделать строго однократным.

3. Работа при высокой нагрузке. Конвейеры данных должны иметь возможность масштабироваться до высокой производительности, необходимой в современных информационных системах и нормально функционировать при внезапных повышениях нагрузки.

Так как Kafka является буфером между производителями и потребителями, то не требуется учитывать производительность обработки информации сторонами. При использовании Kafka не требуется сложный механизм контроля обратного потока данных, поскольку, если производительность производителя превышает производительность потребителя, данные будут накапливаться в Kafka до тех пор, пока потребитель не «догонит» производителя. Возможность Kafka масштабироваться за счет независимого добавления производителей и потребителей делает возможным динамически и независимо масштабировать любую из сторон конвейера, чтобы обеспечить работу при изменении нагрузки.

4. Форматы данных. Одна из важнейших задач конвейеров данных — это согласование их форматов и типов. Различные базы данных и другие системы хранения поддерживают многообразные форматы данных, поэтому внутри системы с конвейером данных возникает необходимость их взаимного согласования.

При использовании Kafka производители и потребители могут применить любой сериализатор (serializer) для представления данных в различных форматах [8]. Также возможно использование фреймворка Kafka Connect [9], включающего типы и схемы данных для автоматического преобразования форматов данных согласно заданной конфигурации. Например, с помощью Kafka Connect можно настроить преобразование данных из MySQL в Snowflake.

5. Порядок преобразований данных. Существует две парадигмы создания конвейеров данных: ETL и ELT.

Конвейер, который извлекает данные, преобразовывает и затем их загружает называется ETL (Extract, transform, load) конвейером — это означает, что конвейер данных отвечает за изменение проходящих через него данных. ETL конвейер дает значительную экономию времени и места, поскольку не требуется сохранять данные, менять их и повторно сохранять. В зависимости от характера необходимых преобразований это преимущество может быть полезным или нежелательным из-за того, что конвейер данных должен отвечать как за вычисления, так и за хранение данных. Основной недостаток такого подхода заключается в том, что производимые в конвейере данных преобразования могут лишить другие блоки обработки возможности обрабатывать эти данные в дальнейшем. Например, создатель конвейера между MongoDB и MySQL решил отфильтровать часть событий и убрать из записей некоторые поля — в этом случае у всех обращающихся к данным в MySQL пользователям и приложениям предоставляется доступ лишь к части данных; если им по-

требуется доступ к отсутствующим полям, придется перестраивать конвейер и повторно обрабатывать данные (если они еще доступны).

Конвейер, который извлекает данные, загружает и затем их преобразовывает называется ELT (Extract, load, transform) конвейером — это означает, что он лишь минимально преобразует данные (в основном это касается преобразования типов данных), чтобы попадающие по месту назначения данные как можно меньше отличались от исходных. Затем целевая система собирает эти «сырые» данные и обрабатывает их должным образом. Преимущество ELT конвейеров заключается в их гибкости — у пользователей целевой системы есть доступ ко всем данным. В этих системах проще производить поиск возникающих при работе проблем, поскольку вся обработка данных выполняется в одной системе, а не распределяется между конвейером и дополнительными приложениями. Недостаток ELT конвейеров заключается в расходе ресурсов процессора и хранилища в целевой системе.

Применение Kafka Connect позволяет использовать функцию преобразования одного сообщения (Single Message Transformation), которая преобразует записи во время их копирования из источника в Kafka или из Kafka в требуемый элемент системы. Она включает в себя маршрутизацию сообщений в различные темы, фильтрацию сообщений, изменение типов данных и редактирование определенных полей. Более сложные преобразования, включающие объединение и агрегирование, можно выполнить с помощью клиентской библиотеки Kafka Streams [10].

6. Безопасность. Основные нюансы безопасности при работе с конвейерами данных:

— гарантии в шифровании проходящих через конвейер данных. Это особенно важно для конвейеров, проходящих через границы центров обработки данных (ЦОД);

— контроль доступа субъектов, которые могут вносить изменения в конвейер;

— обеспечение аутентификации конвейером при чтении им данных из мест с контролируемым доступом;

— соблюдение законов и нормативных актов (той страны, где располагаются ЦОД системы) при работе с персонально идентифицируемой информацией, касающейся ее хранения, использования и доступа к ней;

— контроль прав доступа к данным, поступающим в Kafka.

Kafka предоставляет возможность шифрования данных при передаче, когда она встроена в конвейер между источниками и приемниками данных, поддерживает аутентификацию (через SASL [11]) и авторизацию. В Kafka также имеется журнал аудита для отслеживания санкционированного и несанкционированного доступа. При использовании Kafka Connect все его элементы, включая коннекторы, должны иметь возможность подключения к внешним системам данных и аутентификации в них, при этом в конфигурации коннекторов обязаны присутствовать учетные данные для аутентификации во внешних системах. Хранение учетных данных в конфигурационных файлах нежелательно ввиду сложностей с обращением и доступом к ним — решением этой проблемы является использование внешней системы управления учетными секретами (например, HashiCorp Vault [12]).

7. Обработка сбоев. Правильная обработка и анализ причин сбоев системы очень важны, так как позволяют предотвратить потерю данных в будущем. Сбои в работе могут происходить из-за различных нестандартных событий: дефектные записи, попадающие в конвейер, восстановление работы системы после обработки не поддающихся разбору записей, исправление «плохих» записей (возможно, при вмешательстве оператора) и их повторная обработка.

Брокер сообщений Kafka может быть настроен на долгое хранение всех событий и имеет возможность при необходимости повторно прочитать исходные данные и исправить необходимые ошибки. Это также позволяет воспроизводить события в целевой системе в случае их утери, так как они ранее были сохранены в Kafka.

Одной из важнейших задач при реализации конвейеров данных является расцепление источников и приемников данных. Случайное связывание может возникнуть в следующих случаях:

1. Узкоспециализированные конвейеры. Некоторые компании создают по отдельному конвейеру для каждой пары приложений, которые нужно связать. Например, эти компании могут использовать Logstash [13], чтобы выгрузить журналы в Elasticsearch, и Informatica [14] для перемещения данных из MySQL и XML-файлов в Oracle. Такая практика приводит к сильному связыванию конвейера данных с конкретными конечными точками и образует множество дополнительных точек интеграции, что значительно повышает затраты для развертывания, сопровождения и мониторинга — все это еще больше усложняет введение инноваций, ведь для каждой новой системы, появляющейся в компании, приходится создавать дополнительные конвейеры.

2. Потери метаданных. Если конвейер данных не сохраняет метаданные схемы и не позволяет ей эволюционировать (то есть отсутствует возможность её изменения), производящее данные программное обеспечение оказывается сильно связанным с программным обеспечением, их использующим. Без информации о схеме данных каждый из этих программных продуктов должен будет содержать информацию о способе разбора данных и их интерпретации. Если конвейер поддерживает эволюцию схемы данных, то команды разработчиков могут менять свои приложения независимо друг от друга.

3. Чрезмерная обработка. Определенная обработка данных является неотъемлемым свойством конвейеров, так как данные перемещаются между его различными частями, в которых используются разные форматы и поддерживаются различные сценарии. В том случае, если решения, принятые при создании конвейера, например, о том какие поля сохранять или как производить агрегирование данных, являются чрезмерными, то другие части могут быть ограничены в своих возможностях и для их полноцен-

ного функционирования им необходимо менять начальные настройки (например, одной части системы нужны не агрегированные, а исходные данные для работы). Такое постоянное изменение конвейера в зависимости от смены требований неэффективно, небезопасно и плохо соответствует концепции быстрой адаптации. Для соответствия этой концепции необходимо сохранять как можно больше необработанных данных и разрешать другим блокам системы самим решать, как их обрабатывать и агрегировать.

Использование Kafka во многом позволяет решить проблемы случайного связывания с помощью использования Kafka Connect и выбора правильных коннекторов между приемниками и передатчиками системы с возможностью изменения схем данных.

Выводы

Использование конвейеров данных в крупных системах обработки данных или при их масштабировании в настоящее время является все более предпочтительным вариантом за счет возможностей многоэтапной последовательной обработки различных видов данных без чрезмерного усложнения внутренней логики отдельно взятой системы обработки путем добавления дополнительных функциональных блоков. Производительность СОД, использующей конвейеры данных, напрямую зависит от наличия надежных и эффективных средств взаимодействия между элементами всей системы в целом, особенно между источниками и приемниками данных, что делает необходимым использование правильно выбранного брокера сообщений, который, в свою очередь, должен обладать всеми требуемыми инструментами для интеграции.

Apache Kafka полностью удовлетворяет всем необходимым требованиям для использования в конвейерах данных в системах обработки данных, обладает простой интеграцией с помощью Kafka Connect и является решением с открытым исходным кодом, что позволяет легко реинтегрировать необходимые изменения в будущем.

Литература:

1. Объем и прогноз рынка Интернета вещей (IoT) по продуктам (компоненты, программное обеспечение); по сферам применения — тенденции роста, ключевые игроки, региональный анализ на 2026–2035 гг. // Researchnester: [сайт]. — URL: https://www.researchnester.com/ru/reports/internet-of-things-iot-market/1189/ (дата обращения: 17.11.2025)
2. Что такое конвейер данных? Определение, типы, преимущества и варианты использования // Astera: [сайт]. — URL: https://www.astera.com/ru/type/blog/data-pipeline/ (дата обращения: 17.11.2025)
3. Селезнёв, А. И. Обобщенная модель построения системы обработки данных / А. И. Селезнёв, И. Л. Селезнёв. — Текст: непосредственный // Молодой учёный. — 2024. — № 29. — С. 22–25.
4. Селезнёв, А. И. Брокеры сообщений в системах обработки данных / А. И. Селезнёв, И. Л. Селезнёв. — Текст: непосредственный // Молодой учёный. — 2025. — № 46. — С. 15–19.
5. How to ingest data to Elasticsearch through Kafka // Elastic: [сайт]. — URL: https://www.elastic.co/search-labs/blog/elasticsearch-apache-kafka-ingest-data/ (дата обращения: 17.11.2025)
6. Шапира, Гвен. Apache Kafka. Потоковая обработка и анализ данных / Гвен Шапира, Тодд Палино, Раджини Сиварам, Крит Петти. — 2-е изд. — Санкт-Петербург: ООО «Прогресс книга», 2023. — 512 с. — Текст: непосредственный.
7. Расширенные возможности Apache Kafka // Proselyte: [сайт]. — URL: https://proselyte.net/kafka-extended-abilitites/ (дата обращения: 17.11.2025)
8. Что такое сериализаторы и десериализаторы в Kafka // Kafka-school: [сайт]. — URL: https://kafka-school.ru/blog/kafka-serde/ (дата обращения: 17.11.2025)
9. Introduction to Kafka Connect // Confluent.Developer: [сайт]. — URL: https://developer.confluent.io/courses/kafka-connect/intro/ (дата обращения: 17.11.2025)
10. Kafka Streams для начинающих. Потоковая обработка данных в мире Java // Habr: [сайт]. — URL: https://habr.com/ru/articles/939872/ (дата обращения: 17.11.2025)
11. How to Secure Your Network with the Simple Authentication and Security Layer (SASL) Protocol // Medium: [сайт]. — URL: https://medium.com/@RocketMeUpCybersecurity/how-to-secure-your-network-with-the-simple-authentication-and-security-layer-sasl-protocol-3f00316c77d8/ (дата обращения: 17.11.2025)
12. Secure Kafka with Vault // Hashicorp: [сайт]. — URL: https://developer.hashicorp.com/validated-patterns/vault/vault-securing-kafka/ (дата обращения: 17.11.2025)
13. Для чего нужен Logstash? // Gitverse: [сайт]. — URL: https://gitverse.ru/blog/articles/development/105-dlya-chego-nuzhen-logstash/ (дата обращения: 17.11.2025)
14. What is Informatica? Is It The Data Management Solution You've Been Overlooking // Syncari: [сайт]. — URL: https://syncari.com/blog/what-is-informatica/ (дата обращения: 17.11.2025)