УДК 004.42

Артемов А. В.

студент

Научный руководитель:

Лысакова Татьяна Алексеевна

Старший преподаватель кафедры

инжиниринговых и цифровых технологий

НИУ «БелГУ»

г.Белгород, Россия

АРХИТЕКТУРА ETL-СИСТЕМ ДЛЯ ОБРАБОТКИ ПОТОКОВЫХ ДАННЫХ В РЕАЛЬНОМ ВРЕМЕНИ

В данной статье авторы освещают проблему организации и построения архитектуры ETL-систем для обработки потоковых данных в реальном времени, а также рассматривают основные подходы, технологии и вызовы, возникающие при разработке таких систем в условиях высокой нагрузки и требований к минимальной задержке обработки.

Ключевые слова: потоковые данные; ETL; потоковая архитектура; масштабируемость; отказоустойчивость; Lambda Architecture; Kappa Architecture.

Artemov A. V.

student

Scientific Supervisor:

Lisakova Tatiana Alekseevna

Senior lecturer, department of

Engineering and Digital Technologies

Belgorod State University

ARCHITECTURE OF ETL SYSTEMS FOR REAL-TIME STREAMING DATA PROCESSING

In this article, the authors address the problem of organizing and building the architecture of ETL systems for processing streaming data in real time, and also examine the main approaches, technologies, and challenges that arise when developing such systems under high load conditions and requirements for minimal processing latency.

Keywords: streaming data; ETL; streaming architecture; scalability; fault tolerance; Lambda Architecture; Kappa Architecture.

Потоковые данные представляют собой непрерывный поток информации, который поступает в систему в реальном времени. В отличие от традиционных наборов данных, которые сначала собираются и затем подвергаются обработке, потоковые данные генерируются и передаются без перерывов, что требует мгновенного реагирования со стороны вычислительных систем. Эти данные отличаются высокой скоростью поступления, изменчивостью и часто значительным объемом.

Современные информационные системы все чаще сталкиваются с необходимостью обработки потоковых данных. Это связано с быстрым развитием цифровых технологий и увеличением количества источников данных. Потоковые данные активно используются в различных сферах. Например, в финансовом секторе они служат для анализа транзакций и котировок, в криптовалютной индустрии - для мониторинга торговых операций и событий в блокчейне. В системах интернета вещей они помогают обрабатывать данные, поступающие с датчиков, а также находят применение в телекоммуникациях, логистике и онлайн-сервисах. В социальных сетях потоковые данные используются для анализа поведения пользователей, а в системах безопасности - для обнаружения аномальных событий.

Обработка данных в реальном времени приобретает особую важность, так как позволяет своевременно реагировать на происходящие события. Например, в финансовых системах даже небольшая задержка, составляющая несколько секунд, может привести к значительным убыткам, а в промышленности - к аварийным ситуациям. В связи с этим возникает необходимость разработки архитектур, которые обеспечивают быструю, надежную и масштабируемую обработку потоковых данных.

ETL (Extract, Transform, Load) представляет собой классический процесс обработки данных, состоящий из трех ключевых этапов: извлечение данных, их преобразование и загрузка в целевую систему хранения. На

первом этапе данные извлекаются из различных источников, таких как базы данных, API, файлы или сенсоры. Этап преобразования включает в себя операции очистки, нормализации, агрегации и обогащения данных. Завершающий этап - это загрузка обработанных данных в хранилище, например, в аналитическую базу данных.

Традиционные ETL-системы предназначены для пакетной обработки данных. В данном случае данные накапливаются в течение некоторого времени, после чего обрабатываются пакетами. Такой подход эффективен для аналитических задач, однако он не подходит для ситуаций, где требуется обработка информации в реальном времени.

потоковый ETL является эволюцией классической концепции, адаптированной для работы с данными в реальном времени. В этом подходе этапы извлечения, преобразования и загрузки выполняются непрерывно, по мере поступления данных. Это требует специальной архитектуры системы, обеспечивающей низкие задержки и высокую пропускную способность. В отличие от пакетной обработки, streaming ETL ориентирован не на периодическую обработку, а на постоянный поток событий.

Архитектура систем потокового ETL основывается на нескольких ключевых компонентах, которые обеспечивают полный цикл обработки данных. К основным элементам относятся источники данных, транспортный слой, слой обработки и слой хранения.

Источники данных - это системы, которые генерируют непрерывный поток информации. К ним могут относиться финансовые биржи, датчики IoT, пользовательские приложения, серверные логи и другие. Эти данные передаются через транспортный слой, который, как правило, реализуется с использованием брокеров сообщений. Брокеры сообщений обеспечивают надежную и масштабируемую доставку данных между различными компонентами системы, а также позволяют буферизовать поток данных и эффективно управлять системой нагрузки.

Следующим компонентом является слой потоковой обработки, на котором выполняются операции трансформации данных, такие как фильтрация, агрегация, объединение потоков и вычисление метрик. Для реализации этого слоя применяются специализированные фреймворки, которые обеспечивают обработку данных с минимальными задержками.

Завершающим этапом является загрузка данных в системы хранения. В зависимости от поставленных задач, это могут быть базы данных реального времени, аналитические хранилища или распределенные файловые системы. Ключевым требованием к системе хранения является обеспечение быстрого доступа к данным и эффективная работа с большими объемами информации.

Принцип работы пайплайна заключается в последовательной передаче данных через все компоненты системы. Поток данных начинается с источника и направляется в брокер сообщений, затем поступает в обработчик, где выполняются необходимые преобразования, и, наконец, результат сохраняется в хранилище. Такой подход гарантирует непрерывность обработки и минимизацию задержек.

В рамках архитекуры ETL сервисов активно используются современные технологии. Для передачи данных часто применяется Apache Kafka, который обеспечивает высокую пропускную способность и отказоустойчивость. Для обработки данных применяются такие инструменты, как Apache Spark Streaming и Apache Flink, которые позволяют выполнять сложные вычисления над потоками в реальном времени. В качестве хранилищ могут использоваться как традиционные базы данных, так и специализированные решения, такие как NoSQL-системы.

Несмотря на широкое распространение потоковых ETL-систем, их разработка и эксплуатация сталкиваются с рядом значительных вызовов. Одной из ключевых проблем является необходимость минимизации задержек в процессе обработки данных. В системах реального времени критически важно обеспечить максимально быстрое движение данных через все этапы

обработки, что требует не только оптимизации алгоритмов, но и применения высокоэффективных технологий с минимальной задержкой.

Еще одной важной проблемой является масштабируемость. В условиях увеличения объема потоковых данных, особенно во время пиковых нагрузок, система должна обеспечивать стабильную работу без снижения производительности. Для этого используются распределенные архитектуры и механизмы горизонтального масштабирования, которые позволяют системе эффективно адаптироваться к изменениям нагрузки.

Отказоустойчивость является еще одной критически важной проблемой. В условиях постоянной обработки данных сбои в системе могут привести не только к потере информации, но и к нарушению целостности данных. Поэтому для обеспечения надежности необходимо внедрять механизмы резервирования, репликации и восстановления после сбоев, которые позволяют минимизировать риски потерь и обеспечить бесперебойную работу системы.

Качеству и консистентности данных уделяется особое внимание. В распределенных системах, особенно при высокой скорости обработки, обеспечение строгой согласованности данных становится сложной задачей. В таких условиях возникают проблемы, такие как дублирование событий, потеря данных и нарушение порядка их поступления. Для решения этих проблем применяются различные методы, включая идемпотентные операции и использование контрольных точек.

Существует несколько архитектурных подходов для построения систем обработки потоковых данных. Среди них наиболее известны Lambda Architecture и Kappa Architecture.

Lambda Architecture основывается на разделении системы на два слоя: пакетный и потоковый. Пакетный слой обрабатывает полный объем данных и генерирует точные результаты, тогда как потоковый слой отвечает за быструю обработку новых данных с минимальными задержками. Результаты обоих слоев комбинируются для получения итогового результата. Этот

подход обеспечивает высокую точность и надежность, однако он значительно усложняет архитектуру системы и требует поддержки двух параллельных процессов обработки данных.

Kappa Architecture представляет собой более простой подход, который отказывается от использования пакетного слоя. В этой архитектуре вся обработка данных происходит в режиме потока, и в случае необходимости пересчета данных используется повторное воспроизведение потока. Это решение снижает сложность системы и уменьшает затраты на ее поддержку. Однако, в отличие от Lambda Architecture, оно требует более высокой надежности и производительности от потокового слоя. На рисунке 1 показаны ключевые различия между архитектурами Lambda и Kappa.

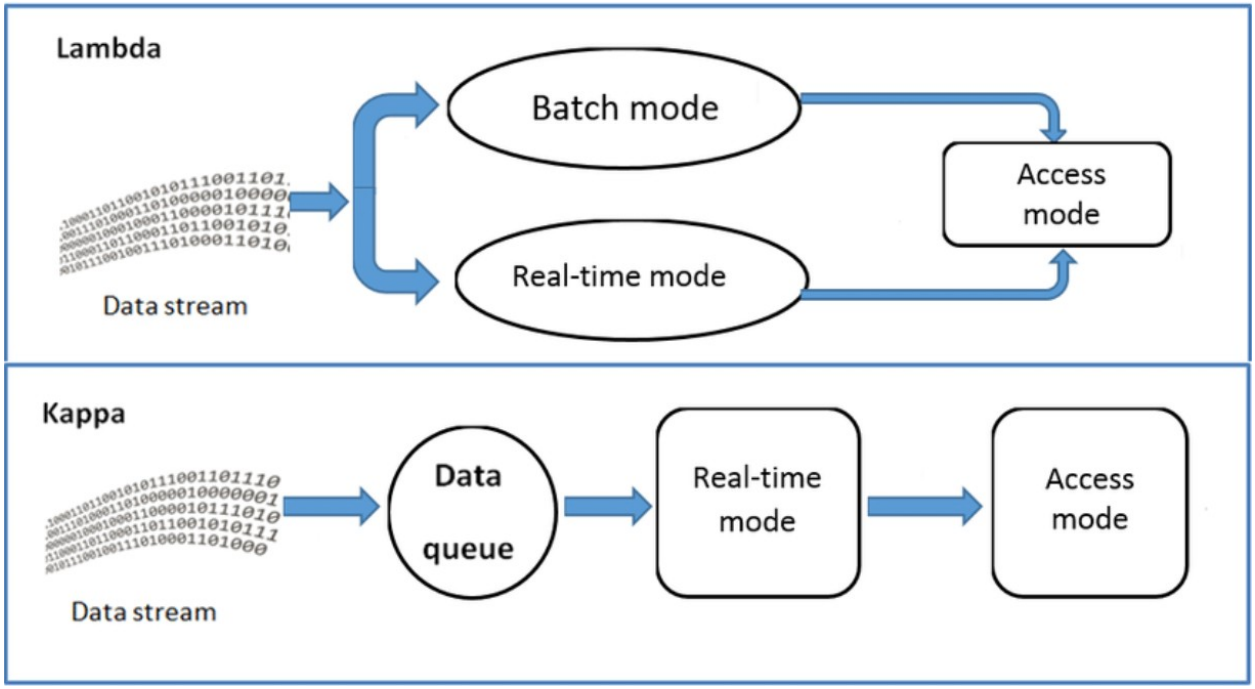

Рисунок 1 - Различия между Lambda и Kappa ахитектурами

Архитектура ETL-систем для обработки потоковых данных в реальном времени занимает центральное место в современных информационных технологиях. Этот сдвиг от пакетной обработки к потоковой связан с необходимостью мгновенной реакции на события и эффективной работы с растущими объемами данных, которые постоянно поступают в систему.

Создание таких систем связано с решением множества сложных задач, включая минимизацию задержек, обеспечение масштабируемости, отказоустойчивости и поддержание консистентности данных. Архитектурные подходы, такие как Lambda и Kappa, предлагают различные способы балансировки этих требований, каждый из которых имеет свои преимущества и ограничения.

Перспективы развития потоковых ETL-систем включают дальнейшее сокращение задержек, интеграцию с технологиями машинного обучения и расширение сфер их применения. В условиях цифровой экономики способность эффективно обрабатывать данные в реальном времени становится важным конкурентным преимуществом.

СПИСОК ЛИТЕРАТУРЫ

1. Потоковые данные [Электронный ресурс]. - URL: https://www.alachisoft.com/ru/foundations/real-time-processing/streaming-data/ (дата обращения: 11.04.2026).
2. Характеристики потоковых данных [Электронный ресурс]. - URL: https://cyberleninka.ru/article/n/issledovanie-harakteristik-potokov-dannyh-generiruemyh-web-serverom (дата обращения: 11.04.2026).
3. Где применяются потоковые данные [Электронный ресурс]. - URL: https://ya.ru/neurum/c/tehnologii/q/v_kakih_otraslyah_primenyaetsya_potokovaya_eec29b61 (дата обращения: 12.04.2026).
4. Что такое ETL [Электронный ресурс]. - URL: https://practicum.yandex.ru/blog/chto-takoe-etl/ (дата обращения: 11.04.2026).
5. Пакетная обработка данных [Электронный ресурс]. - URL: https://aws.amazon.com/ru/what-is/batch-processing/ (дата обращения: 13.04.2026).
6. Потоковая обработка данных [Электронный ресурс]. - URL: https://www.solix.com/ru/kb/stream-processing/ (дата обращения: 11.04.2026).
7. Apache kafka [Электронный ресурс]. - URL: https://ru.wikipedia.org/wiki/Apache_Kafka (дата обращения: 12.04.2026).
8. Виды NoSQL [Электронный ресурс]. - URL: https://yandex.cloud/ru/blog/posts/2022/10/nosql (дата обращения: 12.04.2026).
9. Минимизация задержек [Электронный ресурс]. - URL: https://habr.com/ru/companies/mws/articles/312038/ (дата обращения: 11.04.2026).
10. Масштабируемость [Электронный ресурс]. - URL: https://ru.wikipedia.org/wiki/ (дата обращения: 11.04.2026).

11. Отказоустойчивость [Электронный ресурс]. - URL: https://ytsaurus.tech/docs/ru/user-guide/data-processing/reliability (дата обращения: 13.04.2026).
12. Согласованность данных [Электронный ресурс]. - URL: https://habr.com/ru/companies/vk/articles/723734/ (дата обращения: 13.04.2026).
13. Lambda architecture [Электронный ресурс]. - URL: https://bigdataschool.ru/blog/what-is-lambda-architecture/ (дата обращения: 13.04.2026).
14. Kappa architecture [Электронный ресурс]. - URL: https://habr.com/ru/companies/otus/articles/769350/ (дата обращения: 11.04.2026).