**ОПИСАНИЕ ФУНКЦИОНАЛЬНЫХ ХАРАКТЕРИСТИК**

**RT.STREAMING**

2025

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

| Термин | Определение |
| --- | --- |
| Разработчик | Организация, которая выполняет разработку задач (в том числе анализ требований, проектирование, приёмочные испытания) в процессе жизненного цикла. |
| Пользователь | Лицо или группа лиц, извлекающих пользу из RT.Streaming в процессе его применения. |

ОГЛАВЛЕНИЕ

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 2](#_Toc81486027)

[1 ФУНКЦИОНАЛЬНЫЕ ХАРАКТЕРИСТИКИ 4](#_Toc81486028)

[2 ПРИНЦИПЫ ФУНКЦИОНИРОВАНИЯ RT.STREAMING 6](#_Toc81486029)

[2.1 Структура и функционирование RT.Streaming 6](#_Toc81486030)

[2.1.1 Компонент Kafka 6](#_Toc81486031)

[2.1.2 Компонент NiFi 7](#_Toc81486032)

[2.1.3 Компонент Airflow 9](#_Toc81486033)

[3 БИЗНЕС-ПРОЦЕССЫ 11](#_Toc81486034)

ФУНКЦИОНАЛЬНЫЕ ХАРАКТЕРИСТИКИ

**RT.Streaming** — инструмент для эффективного управления процессами изменения, выгрузки и загрузки данных, позволяющий настроить потоковую или пакетную обработку данных из различных источников.

**RT.Streaming** эффективно решает задачи корпоративной шины обмена данными: она способна получать и обрабатывать данные из многочисленных внешних систем, хранить их в течение нужного для бизнеса периода времени и возвращать потребителям с удобной для них нагрузкой.

**RT.Streaming** обеспечивает реализацию следующих функций:

Обеспечивает безостановочный и отказоустойчивый обмен данными между различными системами компании, позволяет:

строить потоковые конвейеры данных в реальном времени, которые надёжно передают данные между системами или приложениями;

разрабатывать потоковые приложения в реальном времени, которые преобразуют или реагируют на потоки данных;

хранить потоки записей, задавая нужный период хранения данных;

разграничивать права доступа к потокам данных, изолируя критически важную информацию от несанкционированного доступа.

Предоставляет возможности создания событийной архитектуры.

Имеет визуальный интерфейс, позволяющий соединить различные источники и потребителей данных, организовать pipeline и запланировать регулярность его выполнения.

Позволяет организовать планирование сложных последовательностей задач с помощью создания направленных ацикличных графов (DAG) и отслеживать статус выполнения.

Гибкость и масштабируемость системы: всегда можно добавить в кластер новые серверы без остановки работы системы или же исключить ненужные.

**RT.Streaming** обладает следующими особенностями:

1. Техническая поддержка и консультации пользователей на русском языке, полноценный комплект технической и эксплуатационной документации.

Гибкие варианты поставки и развёртывания решения: от самостоятельной установки дистрибутива в инфраструктуре Заказчика до использования готового решения в облачной инфраструктуре.

Совместимость и возможность интеграции с Hadoop и GreenPlum.

Возможность интеграции с различными источниками данных с помощью odbc/jdbc драйверов.

Удобный пользовательский интерфейс для NiFi и Airflow. Простая организация разработки и тестирования.

Множество коннекторов для загрузки и чтения из Kafka.

Управление с помощью RT.ClusterManager:

автоматическая установка кластера, благодаря собственным ansible-скриптам;

мониторинг и отслеживание состояния каждого компонента системы;

возможность гибкого масштабирования.

Поддержка различных процессов загрузки данных:

batch;

realtime;

nrt.

Программный продукт зарегистрирован в реестре отечественного программного обеспечения.

ПРИНЦИПЫ ФУНКЦИОНИРОВАНИЯ RT.STREAMING

Структура и функционирование RT.Streaming

**RT.Streaming** включает в себя следующие функциональные компоненты, предназначенные для решения соответствующих комплексов задач:

1. **Kafka** — платформа распределённой потоковой передачи событий, используемая для высокопроизводительных конвейеров данных, потоковой аналитики, интеграции данных и критически важных приложений.
2. **NiFi** — платформа обработки событий (сообщений), предоставляющая возможности управления потоками данных из разнообразных источников в режиме реального времени с использованием графического интерфейса.
3. **Airflow** — платформа для разработки, планирования и мониторинга рабочих процессов.

Компонент Kafka

Kafka сочетает в себе три ключевые возможности для сквозной потоковой передачи событий:

1. Публикация (запись) и подписка (чтение) потоков событий, включая непрерывный импорт/экспорт данных из других систем.

Надёжное и длительное хранение потоков событий.

Обработка потоков событий по мере их возникновения или ретроспективно.

Kafka — это распределённая система, состоящая из **серверов** и **клиентов**, которые обмениваются данными через сетевой протокол TCP:

1. **Серверы** — Kafka работает как кластер из одного или нескольких серверов, которые могут охватывать несколько центров обработки данных или облачных регионов. Некоторые из этих серверов образуют уровень хранения, называемый брокерами. Другие серверы используют Kafka Connect для непрерывного импорта и экспорта данных в виде потоков событий для интеграции Kafka с существующими системами, такими как реляционные базы данных, а также с другими кластерами Kafka.

**Клиенты** — они позволяют писать распределённые приложения и микросервисы, которые считывают, записывают и обрабатывают потоки событий параллельно, масштабно и отказоустойчивым образом даже в случае сетевых проблем или сбоев машины.

**Событие** фиксирует тот факт, что «что-то произошло». Событие также называется записью или сообщением. Когда вы читаете или записываете данные в Kafka, вы делаете это в форме событий. Концептуально событие имеет ключ, значение, метку времени и необязательные заголовки метаданных.

**Поставщики** — это те клиентские приложения, которые публикуют (записывают) события в Kafka, а **потребители** — это те, которые подписываются (читают и обрабатывают) на эти события. В Kafka поставщики и потребители полностью отделены друг от друга и не зависят друг от друга, что является ключевым элементом дизайна для достижения высокой масштабируемости Kafka.

События организованы и надёжно хранятся в **топиках**. Очень упрощённо, топик похож на папку в файловой системе, а события — это файлы в этой папке. Топики в Kafka могут иметь ноль, одного или нескольких поставщиков, которые записывают в топик события, а также ноль, одного или нескольких потребителей, которые подписываются на эти события. События в топике можно читать сколь угодно часто — в отличие от традиционных систем обмена сообщениями, события не удаляются после использования. Вместо этого вы определяете, как долго Kafka должен сохранять ваши события с помощью настройки конфигурации для каждого топика, после чего старые события будут удалены. Производительность Kafka практически не зависит от размера данных, поэтому хранить данные в течение длительного времени это нормально.

Топики поделены на **партиции**, то есть топик распределён по нескольким «корзинам», расположенным на разных брокерах Kafka. Это распределённое размещение данных очень важно для масштабируемости, потому что оно позволяет клиентским приложениям одновременно читать и записывать данные от многих брокеров. Когда новое событие публикуется в топике, оно фактически добавляется к одной из партиций топика. События с одним и тем же ключом события (например, идентификатор клиента) записываются в одну и ту же партицию, и Kafka гарантирует, что любой потребитель данной партиции топика всегда будет читать события данной партиции в том же порядке, в котором они были записаны.

Чтобы сделать данные отказоустойчивыми и высокодоступными, каждый топик можно **реплицировать**, поэтому всегда будут несколько брокеров, у которых есть копия данных на случай, если что-то пойдёт не так. Общей производственной настройкой является фактор репликации 3, то есть всегда будет три копии данных. Репликация выполняется на уровне топик-партиции.

Компонент NiFi

NiFi предназначен для автоматизации потока данных между системами. В рамках NiFi **поток данных** **(DataFlow)** используется для обозначения автоматизированного и управляемого потока информации между системами.

В NiFi используются следующие фундаментальные концепции:

1. **FlowFile** — сущность, представляющая собой объект с содержимым от нуля и более байт и соответствующих ему атрибутов. Это могут быть как сами данные (например, поток Kafka сообщений), так и результат работы процессора, который не содержит данных как таковых, а лишь атрибуты, сгенерированные в результате выполнения запроса. Атрибуты представляют собой метаданные FlowFile.

**Процессор FlowFile (Processor)** — та сущность, которая выполняет основную работу в NiFi. Процессор, как правило, имеет одну или несколько функций по работе с FlowFile: создание, чтение/запись и изменение содержимого, чтение/запись/изменение атрибутов, маршрутизация.

**Соединение (Connection)** — обеспечивает подключение и передачу FlowFile между различными процессорами и некоторыми другими сущностями NiFi. Соединение помещает FlowFile в очередь, после чего передаёт его далее по цепочке. Можно настроить, как FlowFiles выбираются из очереди, их время жизни, максимальное количество и максимальный размер всех объектов в очереди.

**Контроллер потока (Flow Controller)** — контроллер потока поддерживает информацию о том, как процессы подключаются, и управляет потоками и их распределением, чтобы использовать все процессы. Контроллер потока действует как брокер, облегчая обмен FlowFiles между процессорами.

**Группа процессов (Process Group)** — набор процессоров, их подключений и прочих элементов DataFlow. Представляет собой механизм организации множества компонентов в одну логическую структуру. Позволяет упростить понимание DataFlow. Для получения и отправки данных из группы процессов используются Input/Output Ports (входные и выходные порты).

Данная модель проектирования даёт множество преимуществ, которые помогают NiFi быть очень эффективной платформой для создания мощных и масштабируемых потоков данных. Некоторые из этих преимуществ включают:

1. Хорошо подходит для визуального создания и управления ориентированными графами процессоров.

По своей сути асинхронный, что обеспечивает очень высокую пропускную способность и естественную буферизацию даже при колебаниях обработки и скорости потока.

Предоставляет модель с высокой степенью параллелизма, при этом разработчику не нужно беспокоиться о типичных сложностях параллелизма.

Способствует разработке связанных и слабосвязанных компонентов, которые затем могут быть повторно использованы в других контекстах, и продвигает тестируемые юниты.

Соединения с ограниченными ресурсами делают важные функции, такие как противодавление и сброс давления, очень естественными и интуитивно понятными.

Точки, в которых данные входят в систему и выходят из неё, а также то, как они проходят, хорошо понятны и легко отслеживаются.

NiFi выполняется в JVM в операционной системе хоста. Основные компоненты NiFi на JVM следующие:

1. **Веб-сервер** (**Web Server) —** предоставляет веб-интерфейс и REST API.

**Контроллер потока (Flow Controller)** — мозг операции. Он предоставляет потоки для запуска расширений и управляет расписанием получения расширениями ресурсов для выполнения.

**Расширения (Extensions)** — существуют различные типы расширений NiFi. Ключевым моментом здесь является то, что расширения работают и выполняются в JVM.

**Репозиторий FlowFile (FlowFile Repository)** — это то место, в котором NiFi хранит всю известную ему информацию о каждом существующем в данный момент FlowFile в системе.

**Репозиторий Content (Content Repository**) — репозиторий, в котором находится содержимое всех FlowFile, т.е. сами передаваемые данные.

**Репозиторий Provenance (Provenance Repository)** — содержит историю о каждом FlowFile. Каждый раз, когда с FlowFile происходит какое-либо событие (создание, изменение и т.д.), соответствующая информация заносится в данный репозиторий.

NiFi также может работать в кластере.

Компонент Airflow

Airflow — это платформа для программного создания, планирования и мониторинга **рабочих процессов (workflows)**.

Airflow используется для создания рабочих процессов в виде направленных ациклических **графов (Directed Acyclic Graph, DAG)**.

DAG — это объединение ваших **тасков (tasks)**, которые вы хотите выполнить в строго определённой последовательности по определённому расписанию. При проектировке DAG закладывается набор **операторов (Operators)**, на которых будут построены таски внутри DAG.

Оператор — это сущность, на основании которой создаются инстансы тасков, где описывается, что будет происходить во время исполнения инстанса таска. Airflow поставляется уже с набором операторов, которые готовы к использованию, но можно создавать и свои операторы для своих производственных сред.

Чтобы все инстансы тасков выполнялись в Airflow используется **Планировщик (Scheduler)**. Планировщик позволяет организовать очередь и асинхронное и распределённое выполнение тасков. ПланировщикAirflow выполняет ваши таски на массиве **обработчиков (workers)**, следуя указанным зависимостям.

Служебные программы командной строки упрощают выполнение сложных операций на DAG. Пользовательский интерфейс позволяет легко визуализировать конвейеры (pipelines), работающие в производственной среде, отслеживать прогресс и при необходимости устранять неполадки.

В Airflow используются следующие принципы:

1. Динамичность — конвейеры Airflow имеют конфигурацию кода (Python), что позволяет создавать динамические конвейеры. Это позволяет писать код, который динамически создаёт инстансы конвейеров.

Расширяемость — позволяет определять собственных операторов, исполнителей и расширять библиотеку, чтобы она соответствовала уровню абстракции, подходящему для вашей среды.

Элегантность — конвейеры Airflow тонкие и подробные. Параметризация ваших скриптов встроена в ядро ​​Airflow с использованием мощного механизма шаблонов Jinja.

Масштабируемость — Airflow имеет модульную архитектуру и использует очередь сообщений для оркестрации произвольного количества обработчиков. Airflow можно масштабировать бесконечно.

БИЗНЕС-ПРОЦЕССЫ

**RT.Streaming** может выполнять автоматизацию следующих бизнес-процессов:

1. Обработка платежей и финансовых транзакций в режиме реального времени, например, на биржах, в банках и страховых компаниях.
2. Отслеживание и мониторинг автомобилей, грузовиков, автопарков и грузов в режиме реального времени, например, в логистике и автомобильной промышленности.
3. Непрерывный сбор и анализ данных датчиков с устройств Интернета вещей или другого оборудования, например, на заводах и в ветряных парках.
4. Сбор и немедленное реагирование на взаимодействия и заказы клиентов, например, в розничной торговле, гостиничном и туристическом бизнесе, а также в мобильных приложениях.
5. Наблюдение за пациентами, находящимися в стационаре, и прогнозирование изменений их состояния, чтобы обеспечить своевременное лечение в экстренных случаях.
6. Подключение, хранение и предоставление доступа к данным, произведённым различными подразделениями компании.
7. Обслуживание платформ данных, событийно-ориентированных архитектур и микросервисов.