

Сбор данных

Сбор данных — это первый и один из наиболее критических этапов в конвейере больших данных. На этом этапе данные из различных источников собираются и сохраняются для последующего анализа и обработки. Качество и точность собранных данных напрямую влияют на качество последующего анализа и принимаемых на его основе решений. Ошибки на этапе сбора данных могут привести к существенным проблемам на последующих этапах.

Вот примеры источников данных и методы сбора:

Источник	Подходящий метод сбора
Веб-сайт фронтенд (Действия пользователей на веб-страницах, взаимодействие с интерфейсом, переходы по ссылкам и т.д.)	User Interaction Tracking: Этот метод позволяет отслеживать и анализировать действия пользователей на веб-сайте. Используется для сбора данных о кликах, просмотрах страниц, времени, проведенном на сайте, и других взаимодействиях. Обычно это осуществляется через JavaScript-библиотеки, такие как Google Analytics или Mixpanel.
Бэкенд сервис (Серверные компоненты, которые обрабатывают запросы от клиентов, записи в базу данных, обработка ошибок, логирование и т.д.)	Log Aggregation: Этот метод заключается в сборе и агрегации логов с различных серверов и сервисов. Инструменты вроде ELK Stack или Splunk используются для хранения, индексации и анализа этих логов. Этот метод чрезвычайно полезен для мониторинга производительности, обнаружения ошибок и безопасности.
Файловые системы (Файлы и папки, хранящиеся на серверах или локальных машинах, могут включать в себя документы, изображения, аудио- и видеофайлы и т.д.)	File Transfer: Этот метод подразумевает перенос файлов из одного места в другое. Обычно это делается через FTP, SFTP или другие протоколы передачи файлов. В некоторых случаях используются инструменты для синхронизации файловых систем, такие как rsync.
Базы данных (Реляционные или NoSQL базы данных, которые хранят структурированную или полуструктурированную информацию)	Database Query: Этот метод заключается в выполнении запросов к базе данных для извлечения нужных данных. Запросы, как правило, делаются на языке SQL или его аналогах. Эти данные затем могут быть преобразованы и загружены в другое хранилище для анализа, обычно с использованием ETL-процессов.

Источник	Подходящий метод сбора
API (Программный интерфейс приложения, предоставляющий доступ к определенным функциям или данным)	API Calls: Этот метод предполагает программное взаимодействие с API для извлечения данных. HTTP-запросы отправляются к определенному API, и ответы анализируются для извлечения нужной информации. Это может быть реализовано с помощью различных языков программирования, таких как Python, Java, JavaScript и других.
Стриминговые данные (видео / аудио) (Потоковые мультимедийные данные, часто требуют обработки в реальном времени)	Stream Processing: Этот метод предназначен для обработки данных в реальном времени. Инструменты вроде Apache Kafka или Apache Flink используются для сбора, обработки и анализа данных на лету. В контексте аудио и видео, это может включать в себя анализ контента для автоматической классификации, распознавание образов или даже реальное время анализа.
Сенсоры и устройства IoT (Физические устройства, собирающие различные виды данных, такие как температура, влажность, геолокация и т.д.)	Sensor Data Collection: Этот метод включает в себя сбор данных с физических сенсоров и устройств IoT. Данные, как правило, отправляются на центральный сервер или облачное хранилище для последующего анализа. Протоколы вроде MQTT или CoAP часто используются для надежной и эффективной передачи данных.

Вот некоторые популярные открытые (бесплатные) инструменты для сбора данных, от фирмы Apache:

DistCp (Distributed Copy)

- **Что это:** Это инструмент для копирования больших объёмов данных с одного места в другое.
- **Как работает:** DistCp копирует файлы, распределяя задачу копирования между множеством компьютеров для ускорения процесса.

Sqoop

- **Что это:** Этот инструмент помогает перемещать данные между обычными базами данных (например, MySQL, Oracle) и системами для хранения больших данных.

- **Как работает:** Sqoop имеет встроенные "соединители" для разных типов баз данных. Он автоматически переносит данные из базы данных в систему хранения больших данных или обратно.

Spark Streaming

- **Что это:** Инструмент для обработки данных в реальном времени, которые поступают непрерывно, например, от датчиков или социальных сетей.
- **Как работает:** Он разбивает входящие данные на маленькие "пакеты" и обрабатывает их по одному.

Kafka

- **Что это:** Система для управления потоками данных, которая может принимать данные от различных источников и передавать их различным приемникам.
- **Как работает:** Вы можете отправить данные в Kafka, и они будут храниться там до тех пор, пока другая система или приложение не заберет их.

Flume

- **Что это:** Инструмент для сбора больших объёмов данных, часто в формате логов, для последующего анализа.
- **Как работает:** Flume собирает данные с разных источников и сохраняет их в централизованной системе хранения данных для последующего анализа.

Flink

- **Что это:** Платформа для обработки данных, которая может работать как с постоянно поступающими данными, так и с уже сохранёнными данными.
- **Как работает:** Flink может обрабатывать данные "на лету" или работать с ними после их сохранения, в зависимости от задачи.

Эти инструменты различаются по функциональности и применению, но все они полезны для сбора и обработки больших объёмов данных.

Какие компромиссы может решать архитектор на этапе сбора данных:

1. **Скорость против надежности:** Иногда быстрый сбор данных может привести к ухудшению их надежности. Например, потоковая обработка может быть быстрой, но менее надежной в сравнении с пакетной обработкой.
2. **Стоимость против эффективности:** Более дорогие решения могут предложить лучшую производительность и надежность, но иногда бюджет может ограничивать выбор.
3. **Простота против гибкости:** Простые решения легче внедрить и поддерживать, но они могут быть менее гибкими в долгосрочной перспективе по сравнению с более сложными, но гибкими архитектурами.
4. **Немедленная обработка против пакетной обработки:** Решение о том, обрабатывать данные в реальном времени или накапливать их для пакетной обработки, также является компромиссом между скоростью и надежностью.
5. **Облачные решения против локальных:** Облачные решения предлагают гибкость и масштабируемость, но могут быть менее приемлемы с точки зрения безопасности или соответствия нормативам.