Департамент образования города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

Тяпкина Полина Андреевна

Лабораторная работа по «Инструменты для хранения и обработки больших данных»

Выполнил:
Тяпкина Полина Андреевна
Проверил:
Босенко Тимур Муртазович

Курс обучения: 4
Форма обучения: очная

Москва
2025

Лабораторная работа 3-1. Проектирование архитектуры хранилища больших данных

Цель работы: разработать комплексную архитектуру хранилища больших данных для предложенного бизнес-сценария, обосновать выбор технологического стека и визуализировать потоки данных.

Оборудование и программное обеспечение

- Инструменты для создания диаграмм: draw.io (Diagrams.net), Miro, Lucidchart или аналоги.
- Доступ к сети Интернет для исследования технической документации по современным платформам и инструментам для работы с большими данными.

Краткая теоретическая справка

Архитектура больших данных — это каркас, который описывает сбор, хранение, обработку, анализ и визуализацию больших и сложных наборов данных. Правильно спроектированная архитектура является основой для извлечения ценных инсайтов и принятия решений на основе данных.

- Data Lake (Озеро данных): централизованное хранилище, которое позволяет хранить огромные объемы структурированных, полуструктурированных и неструктурированных данных в их исходном, необработанном формате. Основная идея — собрать все данные в одном месте для последующего анализа.
- Data Warehouse (Хранилище данных, DWH): система, предназначенная для хранения и анализа структурированных данных из различных источников. Данные предварительно очищаются, трансформируются (ETL/ELT) и моделируются для оптимизации аналитических запросов и отчетности.
- Data Lakehouse: современная гибридная архитектура, которая сочетает в себе гибкость и масштабируемость Data Lake с производительностью и возможностями управления данными Data Warehouse. Она позволяет выполнять BI-запросы и задачи

машинного обучения непосредственно на данных в озере, используя такие форматы, как Apache Iceberg, Delta Lake или Apache Hudi.

Компоненты архитектуры:

- Слой сбора данных (Ingestion): Apache Kafka, Confluent, Amazon Kinesis, Vector, Airbyte, Fivetran.
- Слой хранения (Storage): облачные хранилища (Amazon S3, Google Cloud Storage), форматы таблиц для Data Lakehouse (Delta Lake, Apache Iceberg, Hudi).
- Слой обработки (Processing): Apache Spark, Apache Flink, облачные платформы (Databricks, Snowflake, Google BigQuery), аналитические СУБД (ClickHouse, Greenplum).
- Слой аналитики и визуализации (Analytics & Visualization): Tableau, Power BI, Metabase, Looker, Apache Superset, Grafana.
- Слой оркестрации (Orchestration): Apache Airflow, Dagster, Prefect.
- Управление данными (Data Governance): Apache Atlas, Amundsen, OpenMetadata.

Задание для самостоятельной работы

Задача: платформа "Умного города": управление городской инфраструктурой (светофоры, освещение), анализ транспортных потоков, мониторинг экологической обстановки. Источники: данные с дорожных камер, датчиков качества воздуха, общественного транспорта.

Цель работы: разработать комплексную архитектуру хранилища больших данных для платформы "Умного города", обеспечивающую эффективное управление городской инфраструктурой, анализ транспортных потоков и мониторинг экологической обстановки. Архитектура должна обеспечивать надежное хранение, обработку и анализ больших объемов разнородных данных, а также поддерживать современные требования к безопасности, масштабируемости и отказоустойчивости.

1. Анализ требований

1. Анализ требований к платформе «Умный город»

1.1 Объем данных

- Ожидаемый объем: 50-100 ТБ в год - мало.
- Рост объема данных: 30-50% ежегодно, связанный с расширением городской инфраструктуры и увеличением числа датчиков.

1.2 Скорость получения данных

- Потоковые данные с дорожных камер и датчиков: в режиме реального времени, до нескольких тысяч событий в секунду.
- Обновления данных с общественного транспорта (GPS, расписания): с интервалом от нескольких секунд до минут.
- Периодические данные с датчиков воздуха и других систем: обновления каждые 5-15 минут.

1.3 Типы данных

- Структурированные: телеметрия транспорта, управление светофорами и освещением (около 30%).

- Полуструктурированные: данные с датчиков качества воздуха, JSON-форматы мониторинговых систем (около 50%).
- Неструктурированные: потоковое видео и изображения с камер, аудио и другие сенсорные данные (около 20%).

1.4 Требования к обработке

- Аналитика в реальном времени для управления инфраструктурой (светофорами, освещением).
- Пакетный анализ транспортных потоков и экологических показателей — еженедельно и ежемесячно.
- Прогнозирование загруженности дорог и экологической обстановки — ежеквартально.
- Обучение моделей машинного обучения для предсказания транспортных заторов и аномалий экологии.
- Создание оперативных отчетов и дашбордов для управленцев в режиме реального времени.

1.5 Доступность данных

- Время отклика для аналитических запросов: менее 30 секунд.
- Доступность системы: не ниже 99.9%, что допускает максимум около 8,8 часов простоя в год.

1.6 Безопасность данных

- Шифрование данных в состоянии покоя и при передаче.
- Многофакторная аутентификация для доступа к системам и данным.
- Ведение аудита всех операций с данными.
- Обеспечение соответствия российскому законодательству, включая требования 152-ФЗ «О персональных данных».

Источники данных:

- Дорожные камеры (потоковое видео/изображения, структурированные данные с распознавания)

- Датчики качества воздуха (потоковые и периодические значения, полуструктурированные данные)
- Общественный транспорт (GPS-трекинг, расписание, структурированные данные)

Типы данных:

- Структурированные: телеметрия транспорта, светофоров, освещения
- Полуструктурированные: данные с датчиков, JSON из систем мониторинга
- Потоковые данные: видео и сенсорные данные в реальном времени

Объемы и скорость:

- Высокая скорость поступления данных с камер и датчиков (потоковые данные)
- Ежесуточное накопление десятков-тепрограмм терабайт данных с видео и телеметрии

Бизнес-цели:

- Аналитика в реальном времени: управление светофорами для оптимизации движения
- Пакетная аналитика: анализ загруженности дорог, прогнозы трафика
- Мониторинг экологии и оперативные оповещения
- Визуализация и дашборды для операторов и управленцев
- Возможность обучения моделей машинного обучения (например, предсказание транспортных заторов)

2. Выбор компонентов архитектуры и технологического стека

Сбор данных (Ingestion)

- Apache Kafka работает как распределённая система обмена сообщениями. Источники данных (например, камеры, датчики) публикуют сообщения в топики Kafka. Эти сообщения сохраняются в последовательных журналах (логах) на дисках брокеров Kafka. Потребители (например, системы обработки данных) подписываются на топики и читают сообщения асинхронно, что позволяет обрабатывать потоки данных с высокой пропускной способностью и гарантировать доставку. Kafka обеспечивает масштабируемость за счет партиционирования топиков и поддерживает отказоустойчивость за счет репликации данных между брокерами.
    - Для чего: обеспечивает надежную передачу и буферизацию больших потоков данных с разных источников.
    - Польза: высокая пропускная способность, масштабируемость, гарантии доставки сообщений.
    - Плюсы: широкое сообщество, устойчивость, интеграция с множеством систем.
    - Минусы: чложность настройки и управления при больших масштабах, требует ресурсов.
- Vector — это агент для сбора данных, который в фоновом режиме захватывает логи и метрики с различных источников (файлы, сетевые протоколы, API), преобразует их и отправляет дальше в систему хранения или анализа. Vector работает, используя конвейеры обработки, которые могут фильтровать, форматировать и агрегировать данные перед отправкой. Он оптимизирован для высокой пропускной способности и низкой задержки, позволяя легко интегрироваться в разные части архитектуры.
    - Для чего: Сбор разнообразных данных с различных источников и отправка их в системы хранения и анализа.
    - Польза: Легкая интеграция, настройка конвейеров обработки данных без большого кода.
    - Плюсы: Высокая производительность, поддержка множества протоколов и форматов.

- Минусы: относительно новая технология, меньшая экосистема по сравнению с Kafka.

- Airbyte — платформа для интеграции данных, которая работает по принципу коннекторов. Она использует готовые или кастомные коннекторы для подключения к источникам (базы данных, сервисы) и вывода данных в целевые системы (хранилища, аналитические платформы). Airbyte автоматизирует процессы извлечения (Extract) и загрузки (Load), синхронизируя данные с минимальными настройками и позволяя регулярно обновлять информацию.
  - Для чего: быстрое подключение внешних источников данных, синхронизация в хранилище.
  - Польза: ускоряет интеграцию источников, минимизирует написание кастомного кода.
  - Плюсы: открытый код, множество готовых коннекторов, удобный интерфейс.
  - Минусы: может требовать доработок для нестандартных систем.

Хранение (Storage)

- MinIO — это объектное хранилище, работающее по принципу распределенного хранения данных с поддержкой API. Данные разбиваются на объекты и сохраняются с избыточностью для надежности. MinIO поддерживает высокую параллельность операций, масштабируемость и обеспечивает быстрый доступ через RESTful API. Благодаря локальному развертыванию минимизируются задержки и управление данными находится под полным контролем.
  - Для чего: хранение больших объемов необработанных и архивных данных.
  - Польза: дешевле облачных решений, удобство локального развертывания и управления, отсутствие ограничений географического доступа.
  - Плюсы: высокая производительность, поддержка масштабирования, простота интеграции.
  - Минусы: требует администрирования, ограничена экосистема по сравнению с крупными облачными провайдерами.

- Delta Lake строится поверх распределенного хранилища (например, MinIO) и реализует управление версиями и транзакциями на уровне файлов данных. Она поддерживает ACID-транзакции, что гарантирует целостность данных при одновременной загрузке и обработке. Delta Lake образует "озеро данных с управлением", позволяя эффективно проводить запросы, обновления и откаты изменений, а также управлять метаданными.
    - Для чего: обеспечение надежного хранения и управления структурированными и полуструктурированными данными.
    - Польза: гарантирует согласованность данных, поддерживает версионирование и транзакции.
    - Плюсы: интеграция с Apache Spark, повышение качества данных и производительности аналитики.
    - Минусы: сложность внедрения и поддержки требует квалифицированных специалистов.

Обработка (Processing)

- Apache Spark — распределённый движок для пакетной и интерактивной обработки больших данных. Он разбивает задачи на маленькие подзадачи (задачи работы с RDD или DataFrame), которые выполняются параллельно в кластере. Spark обрабатывает данные из хранилищ (например, Delta Lake), используя память для ускорения вычислений, и позволяет строить сложные аналитические модели и трансформации.
    - Для чего: обработка больших объемов данных, сложные аналитические и ETL задачи.
    - Польза: высокая скорость расчетов, поддержка различных форматов данных и языков программирования.
    - Плюсы: широкое сообщество, интеграция с множеством инструментов, удобство написания аналитики.
    - Минусы: ресурсоемкость кластера, требует настройки и оптимизации.
- Apache Flink фокусируется на потоковой обработке данных с малой задержкой. Он работает с непрерывными потоками входящих событий,

поддерживает состояние и окна времени для агрегирования данных. Flink гарантирует точность обработки (exactly-once semantics) и обеспечивает масштабируемость за счет распределенной обработки потоков в кластере.

- Для чего: анализ данных в реальном времени, реагирование на события.
- Польза: минимальная задержка обработки, поддержка сложной логики обработки потоков.
- Плюсы: высокая производительность в реальном времени, точность и гарантии обработки.
- Минусы: более высокая сложность разработки потоковых приложений по сравнению со Spark.

Аналитика и машинное обучение

- Jupyter Notebooks — интерактивная среда, которая запускает код на сервере (чаще всего Python). Пользователь пишет, запускает и визуализирует результаты в браузере по блокам. Это удобно для исследовательского анализа и разработки моделей, где можно экспериментировать, видеть результаты и документировать процесс.
  - Для чего: исследовательская аналитика, разработка и тестирование моделей машинного обучения.
  - Польза: удобный интерфейс для визуализации и кодирования в одном месте.
  - Плюсы: популярность, богатая экосистема, поддержка множества библиотек.
  - Минусы: может быть неэффективным для продакшен сред и больших объемов данных.
- TensorFlow — фреймворк, который строит вычислительный граф для машинного обучения. Он разбивает задачи обучения моделей на операции в графе, которые выполняются в распределенной среде на CPU/GPU. Пользователь создает модель, обучает её на данных, а фреймворк оптимизирует вычисления и управляет памятью.

    - Для чего: разработка сложных моделей прогнозирования и анализа.
    - Польза: хорошо масштабируется, поддерживает распределенное обучение.
    - Плюсы: большое сообщество, интеграция с другими инструментами Google, обширная документация.
    - Минусы: крутая кривая обучения, сложность в настройке.
- Metabase работает как веб-приложение, подключенное к базам данных или хранилищам, где хранится аналитика. Пользователи могут создавать запросы и визуализации через интерфейс без программирования. Metabase генерирует SQL-запросы и отображает результаты в виде графиков и таблиц, предоставляя удобные дашборды.
    - Для чего: предоставление удобных дашбордов и отчетов для аналитиков и управленцев.
    - Польза: легкий в установке и использовании, поддержка SQL и NoSQL источников.
    - Плюсы: бесплатный, открытый исходный код, удобный интерфейс.
    - Минусы: ограниченная функциональность по сравнению с коммерческими BI-системами.

Оркестрация и мониторинг

- Airflow организует выполнение задач в Directed Acyclic Graph (DAG). Задачи — это действия вроде загрузки, обработки данных. Airflow планирует и запускает задачи согласно расписанию или зависимости, ведет логирование и мониторинг выполнения, позволяет управлять ошибками и перезапускать процессы.
    - Для чего: автоматизация и упорядочивание задач обработки и интеграции данных.
    - Польза: гибкое управление зависимостями, удобный интерфейс мониторинга.

- Плюсы: распространенность, масштабируемость, поддержка расширений.
- Минусы: сложность настройки многокластерных сред, требует ресурсов.

Управление данными

- Apache Atlas собирает и хранит метаданные о данных, потоках, их происхождении и структуре. Он позволяет отслеживать, кем и как используются данные, помогает контролировать качество и соответствие политиким. Метаданные интегрируются со всеми компонентами архитектуры для централизованного управления.
  - Для чего: обеспечение прозрачности, соответствия политикам и управление качеством данных.
  - Польза: упрощает поиск данных, контроль доступа и анализ происхождения данных.
  - Плюсы: гибкая интеграция, поддержка различных форматов метаданных.
  - Минусы: сложность внедрения и поддержки.

3. Проектирование архитектуры и потоков данных

- Данные с дорожных камер, датчиков воздуха и транспорта поступают в систему через слой сбора — Kafka и Vector.
- Потоковые данные сразу передаются в слой обработки на Flink для анализа в реальном времени (например, изменение сигналов светофоров).
- Данные также сохраняются в Delta Lake на S3 для дальнейшей пакетной обработки Apache Spark (например, анализ трендов и построение отчетов).
- Результаты аналитики и агрегированные данные сохраняются в DWH-слое (оптимизированный для BI).

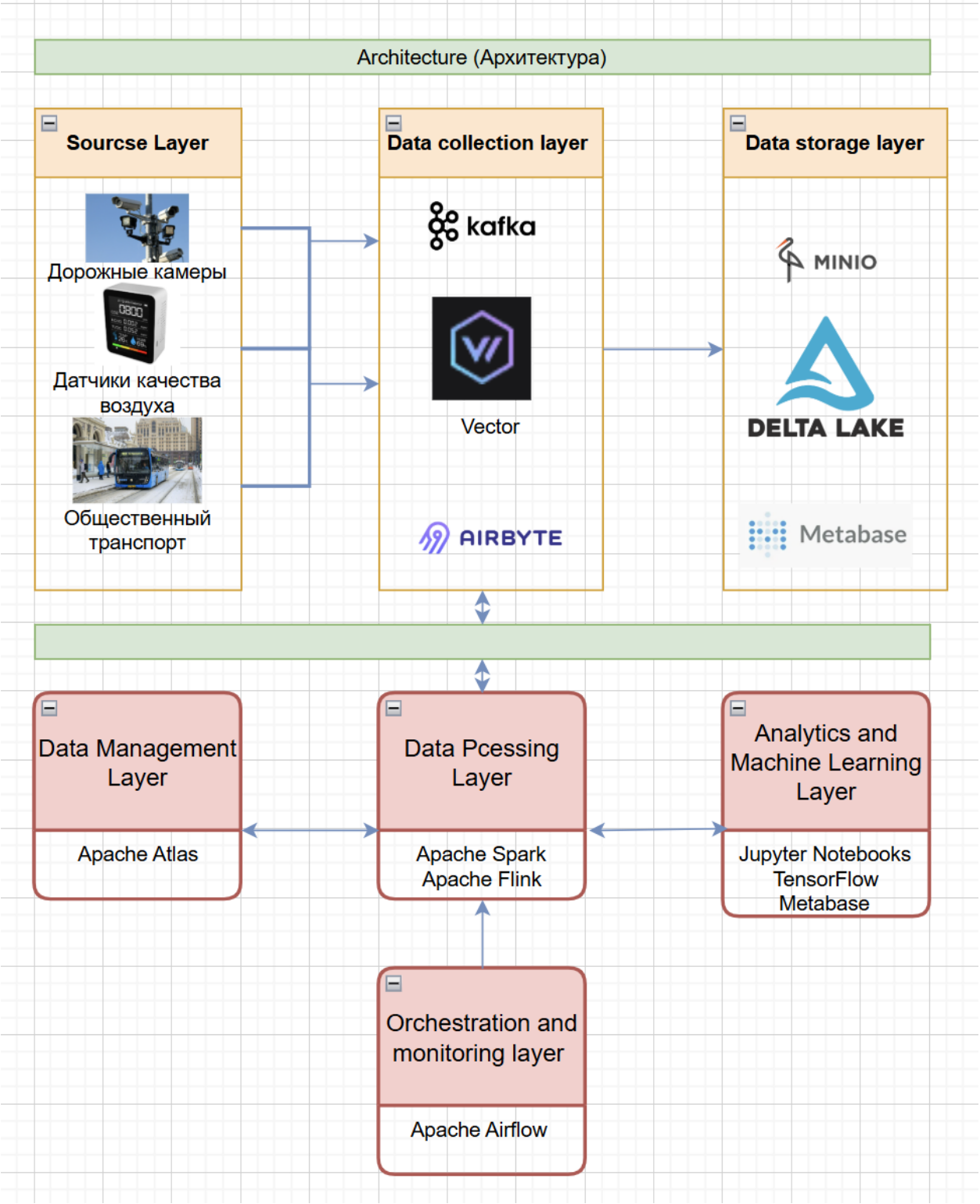
- Аналитики и операторы получают доступ к дашбордам через Grafana и Power BI.
- Оркестрация ETL-процессов обеспечивается Airflow.
- Безопасность реализована через шифрование данных, Kerberos-аутентификацию и контроль доступа Apache Ranger.

4. Масштабирование и отказоустойчивость

- Горизонтальное масштабирование Kafka, Spark и Flink кластерами.
- Репликация данных Delta Lake с multi-AZ (Availability Zone).
- Автоматическое масштабирование ETL и аналитических процессов с Kubernetes (если применимо).
- Резервное копирование данных и аварийное восстановление.

5. Потенциальные проблемы и решения

- Узкое место 1: высокая нагрузка на потоковую обработку данных из камер.
    - Решение: Использование Apache Flink с выделенными ресурсами и оптимизацией топологии потоков.
- Узкое место 3: обеспечение безопасности и конфиденциальности данных граждан.
    - Решение: Полное шифрование, аудит доступа, многофакторная аутентификация и соответствие нормативам защиты данных.


Architecture (Архитектура)
Sourcse Layer
Дорожные камеры
Датчики качества воздуха
Общественный транспорт
Data collection layer
kafka
Vector
AIRBYTE
Data storage layer
MINIO
DELTA LAKE
Metabase
Data Management Layer
Apache Atlas
Data Pcessing Layer
Apache Spark
Apache Flink
Analytics and Machine Learning Layer
Jupyter Notebooks
TensorFlow
Metabase
Orchestration and monitoring layer
Apache Airflow

Итоги и выводы

В ходе лабораторной работы была разработана и проанализирована комплексная архитектура хранилища больших данных, адаптированная под задачи управления городской инфраструктурой, анализа транспортных потоков и мониторинга экологической обстановки.

Ключевые итоги:

- Проведен детальный анализ источников данных, типов, объемов и требований к скорости поступления и обработке, что позволило четко определить архитектурные потребности.
- Выбран современный и сбалансированный стек технологий с использованием Apache Kafka и Vector для надежного сбора потоковых данных, Delta Lake для хранения в формате Data Lakehouse, а также Apache Flink и Apache Spark для обработки в реальном времени и пакетной аналитики.
- Спроектирована архитектура, обеспечивающая целостность данных, масштабируемость и отказоустойчивость за счет горизонтального масштабирования, репликации и облачных решений.
- Организованы средства визуализации и оркестрации (Grafana, Power BI, Apache Airflow), что обеспечивает удобство мониторинга и управления рабочими процессами.
- Внедрены меры по обеспечению безопасности, включая контроль доступа, шифрование данных и аудит, что критично при работе с чувствительной городской информацией.
- Выявлены и проработаны потенциальные узкие места системы с предложениями по их устранению — например, оптимизация потоковой обработки, организация архивного хранения и усиление безопасности.
- Архитектура позволяет добиться бизнес-целей проекта: оперативное управление дорожным движением, мониторинг и реагирование на изменения в экологической обстановке, а также поддержку принятия решений на основе данных и внедрение машинного обучения.

Вывод: Разработанная архитектура хранилища больших данных является надежной, масштабируемой и функционально полной платформой для реализации «Умного города». Она обеспечивает эффективное управление различными источниками данных, высокую

производительность аналитических процессов и безопасность, что соответствует современным требованиям к городским IT-системам.