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

Отчёт по лабораторной работе № 3.1

«Проектирование архитектуры хранилища больших данных»

Выполнила:
студентка группы АДЭУ-211,
Макарова Екатерина Павловна

Преподаватель:
Босенко Тимур Муртазович,
Доцент, кандидат технических наук

Москва 2024

Цель работы: разработать архитектуру хранилища больших данных для заданного сценария использования.

Задача: создать архитектуру хранилища больших данных для средней страховой компании.

1. Требования к данным для средней страховой компании в России

1.1. Объем данных

- Ожидаемый объём 80 ТБ в год.

- Рост 30% ежегодно.

1.2. Скорость получения данных

- Веб-сайт и мобильные приложения: в режиме реального времени до 500 транзакций в секунду.

- Социальные сети: об

1.3. Типы данных

- Структурированные: транзакционные данные, данные CRM, данные СМЭВ 4 (REST через OpenAPI) (75%).

- Полуструктурированные: логи веб-сайтов и приложения, данные XML/ JSON, СМЭВ 3 (XML)(20%).

- Неструктурированные: текстовые отзывы, посты в социальных сетях, финансовые и страховые агрегаторы (5%).

1.4. Требования к обработке

- Оценка страховых рисков: в режиме реального времени (при оформлении страхования).

- Выявление мошенничества: в режиме реального времени (сканирование каждой заявки).

1.5. Доступность данных

- Время отклика <15 секунд

- Доступность системы: 99.95%

1.6. Безопасность данных

- Шифрование данных в состоянии покоя и при передаче

- Строгое соответствие 152-ФЗ и требованиям регулятора.

2. Архитектура хранилища больших данных

2.1. Компоненты архитектуры

Источники данных

- Веб-сайт и мобильные приложения.
- Социальные сети.
- CRM системы
- СМЭВ
- Внешние API (финансовые и страховые агрегаторы, агрегирование данных о погоде, экономических показателях и других факторах, влияющих на страховые риски).

Слой сбора данных

- Apache Kafka: для обработки потоковых данных с высокими показателями (до 500 транзакций в секунду) от веб-сайтов и мобильных приложений.
- Logstash: для сбора и обработки логов с веб-сайтов и приложений.
- Apache NiFi: для автоматизированного сбора данных из различных источников, включая REST API и внешние сервисы.

Обоснование**:** Kafka позволяет обрабатывать до 500 транзакций в секунду и гарантирует высокую доступность и отказоустойчивость. Logstash обеспечивает гибкий сбор и фильтрацию данных, Apache NiFi позволяет легко управлять потоками данных и поддерживает множество форматов.

Слой хранения данных

- PostgreSQL: хранение данных о транзакциях (структурированные данные)
- MongoDB**:** для полуструктурированных данных из социальных сетей
-Amazon S3: для неструктурированных данных
- Keycloak: для персональных данных о клиентах
- Apache Hadoop**:** внешние API

Обоснование:

PostgreSQL: Высокая производительность, поддержка различных типов данных и транзакционной целостности.

MongoDB: Идеальна для хранения полуструктурированных данных, обеспечивая гибкость.

S3: подходит для масштабируемого хранения больших объемов неструктурированных данных.

Keycloak: Гибкая интеграция с поставщиками удостоверений: обеспечивает простую интеграцию с несколькими поставщиками удостоверений (IDP), позволяя организациям использовать существующую инфраструктуру удостоверений.

Слой обработки данных

- Apache Spark для пакетной и потоковой обработки.
- Apache Flink для обработки в реальном времени.

Обоснование: Spark поддерживает распределенную обработку больших объемов данных и может легко интегрироваться с Hadoop для оптимизации работы с большими данными. Flink позволяет обрабатывать потоковые данные с минимальной задержкой, что критично для выявления мошенничества.

Слой аналитики и машинного обучения

- Apache Spark MLlib для машинного обучения
- Power BI для визуализации данных

Обоснование: MLlib предоставляет мощные инструменты для анализа и построения моделей, а BI-инструменты позволяют быстро визуализировать данные и строить отчеты по оценке рисков.

Слой управления данными

- Apache Atlas для управления метаданными.
- Apache Ranger для контроля доступа и аудита.

Обоснование: Apache Atlas позволяет централизованно управлять метаданными и их изменениями, обеспечивая легкость в поиске и управлении

данными. Apache Ranger предоставляет контроль доступа и шифрование, что соответствует требованиям 152-ФЗ.

Слой оркестрации и мониторинга

- Apache Airflow для оркестрации
- Grafana для мониторинга.

Обоснование: Airflow идеально подходит для планирования и управления ETL-заданиями, позволяя следить за графиками выполнения, а Grafana обеспечивает систему мониторинга для отслеживания состояния системы и производительности.

В контексте страховой компании необходимо использовать архитектурный стиль – **«звезда»**, т.к он прост и интуитивно понятен, имеет высокая производительность запросов, так как требуется меньше соединений между таблицами. Высокая производительность важна, т.к нужно оценивать финансовые риски при оформлении страховых транзакций.