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

ДИСЦИПЛИНА:
Инструменты для хранения и обработки больших данных

Лабораторная работа № 3.1

Тема:

«Проектирование архитектуры хранилища больших данных»

Выполнил(а):
Вереина М.С., АДЭУ-211
Преподаватель: Босенко Т.М.

Москва
2024

Лабораторная работа 3-1. Проектирование архитектуры хранилища больших данных

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

Задача: Разработать архитектуру хранилища больших данных для компании, основываясь на предоставленных требованиях. Описать компоненты архитектуры, обосновать выбор технологий и предложить схему потока данных. Для создания диаграммы архитектуры студенты можно использовать draw.io, которое доступно как онлайн-инструмент и как приложение.

Вариант 3. Телекоммуникационная компания

- Объем данных: 1 ПБ в год, рост 40% ежегодно.
- Скорость получения: до 10000 событий в секунду.
- Типы данных: 40% структурированные, 50% полуструктурированные, 10% неструктурированные.
- Требования к обработке: анализ качества связи, прогнозирование нагрузки на сеть.
- Доступность: 99.99%, время отклика <10 секунд.
- Безопасность: шифрование, соответствие 152-ФЗ и отраслевым стандартам.

Алгоритм выполнения работы

1. Анализ требований:

1.1 Объем данных

Ожидаемый объем: 1 ПБ в год

Рост: 40% ежегодно.

1.2 Определить источники данных.

Информация о поведении пользователей, их транзакциях, участии в маркетинговых активностях, типах и объемах потребления трафика и услуг, поступающая со скоростью до 10000 событий в секунду

1.3 Выявить типы данных (структурированные, полуструктурированные, неструктурированные).

Структурированные (из CRM-системы/веб-сайта), полуструктурированные (система мониторинга, платежная система, веб-сайт, CRM-система), неструктурированные (соцсети, веб-сайт, CRM-система).

1.4 Оценить объемы данных и скорость их поступления.

Объем данных: 1 ПБ в год.

Скорость получения: до 10000 событий в секунду.

1.5. Определить требования к аналитике и отчетности.

- анализ качества связи: еженедельно.

- прогнозирование нагрузки на сеть: ежеквартально.

1.6 Доступность данных

- Доступность: 99.99%, время отклика <10 секунд.

2. Архитектура хранилища больших данных

2.1 Компоненты архитектуры

- Для каждого компонента архитектуры предоставить краткое описание его роли и функций.

- Объяснить причины выбора конкретных технологий и компонентов.

1) Источники данных

Структурированные:

1. Данные о вызовах
   - Формат: CSV или JSON
   - Скорость поступления: в реальном времени
   - Источник: CRM-система
2. Данные о счетах и платежах
   - Формат: SQL или CSV
   - Скорость поступления: Пакетная обработка (раз в день или неделю)
   - Источник: CRM-система
3. Данные об абонентах
   - Формат: SQL или CSV
   - Скорость поступления: Пакетная обработка (в реальном времени при изменениях)
   - Источник: CRM-система или веб-сайт
4. Данные о SMS-рассылках
   - Формат: CSV или JSON
   - Скорость поступления: Пакетная обработка (раз в несколько минут)
   - Источник: CRM-система

Неструктурированные:

1. Отзывы и комментарии пользователей

- Формат: Текст
- Скорость поступления: в реальном времени
- Источник: Социальные сети, веб-сайт

2. Обращения в службу поддержки
- Формат: Текстовые сообщения или письма
- Скорость поступления: в реальном времени или пакетная обработка (раз в день)
- Источник: CRM-система

3. Медиа-контент (фото, видео)
- Формат: Изображения (JPEG, PNG) или видео (MP4)
- Скорость поступления: в реальном времени
- Источник: Социальные сети

Полуструктурированные:

1. Данные о транзакциях

- Формат: XML или JSON
- Скорость поступления: в реальном времени
- Источник: Платежные системы

2. Данные о маркетинговых кампаниях

- Формат: JSON или XML
- Скорость поступления: пакетная обработка (раз в день)
- Источник: CRM-системы

3. Логи веб-сайтов и приложений

- Формат: CSV или JSON
- Скорость поступления: в реальном времени
- Источник: веб-сайты

4. Данные об использовании интернета
- Формат: JSON или XML
- Скорость поступления: в реальном времени (обновление раз в несколько секунд)
- Источник: Системы мониторинга сети

5. Данные о технических сбоях
- Формат: JSON или XML
- Скорость поступления: в реальном времени
- Источник: Система мониторинга сети

6. Данные о геолокации устройств
- Формат: JSON
- Скорость поступления: в реальном времени (обновление каждые 5-15 секунд)

- Источник: Система мониторинга сети

2) Слой сбора данных

Компоненты: Logstash, Apache Kafka, SNMP Collector на Zabbix (вспомогательно для сбора).

Роль и функции. Сбор и передача данных от источников в систему обработки. Kafka обеспечивает высокую пропускную способность, а SNMP Collector собирает данные о состоянии сетевых устройств и передает их с помощью Zabbix. Logstash для сбора логов.

Причины выбора. Kafka позволяет обрабатывать потоки данных в реальном времени, что необходимо для телекоммуникационных систем, а SNMP Collector обеспечивает интеграцию с сетевыми устройствами. Logstash является одним из самых популярных и доступных сервисов для сбора логов.

3) Слой сбора данных

Компоненты: Greenplum для структурированных и полуструктурированных данных; Cassandra для неструктурированных.

Роль и функции. Хранение больших объемов данных с возможностью быстрого доступа и анализа.

Причины выбора. Greenplum предназначена для хранения и обработки больших объемов данных. Массово-параллельная архитектура Greenplum и мощные алгоритмы оптимизации подходят для быстрой обработки «тяжёлых» аналитических запросов при работе с многотерабайтными массивами данных. В данной компании ожидается объем данных в 1 ПБ, основная масса которых является структурированными и полуструктурированными. Cassandra же обеспечивает масштабируемость для неструктурированных данных, а так как их объем небольшой, ее можно подключить и для сбора полуструктурированных данных, так как их объем составляет большую часть всех данных.

4) Слой обработки данных

Компоненты: Apache Spark, Apache Flink.

Роль и функции. Обработка и анализ данных в реальном времени.

Причины выбора. Использование обоих фреймворков позволяет комбинировать их сильные стороны: Flink может обрабатывать данные в реальном времени, обеспечивая быструю реакцию на события. Spark может использоваться для анализа больших объемов данных, которые были собраны и обработаны Flink. Оба фреймворка могут интегрироваться с Apache Kafka, что обеспечивает эффективный обмен данными между ними. Совместное использование Apache Spark и Apache Flink позволяет

телекоммуникационной компании эффективно обрабатывать как потоковые, так и пакетные данные, обеспечивая высокую производительность и гибкость архитектуры.

5) Слой аналитики и машинного обучения

Компоненты: TensorFlow, Power BI, PyCharm.

Роль и функции. Создание моделей для прогнозирования нагрузки на сеть и визуализация результатов анализа.

Причины выбора. TensorFlow подходит для сложных вычислений, а Power BI позволяет легко визуализировать данные и является доступным для установки. Для анализа данных и машинного обучения можно использовать библиотеки Python в PyCharm (например, Pandas, NumPy, Scikit-learn). С помощью плагина Google Cloud Code в PyCharm можно строить и запускать приложения как в локальном кластере Kubernetes

6) Слой управления данными

Компоненты: Apache Atlas, Apache NiFi.

Роль и функции. Управление метаданными и потоками данных.

Причины выбора. Используя NiFi, можно собирать данные и передавать их в Atlas для обогащения метаданными. Например, NiFi может отправлять информацию о потоках данных (например, их источниках и типах) в Atlas, что позволяет отслеживать происхождение данных и их изменения на протяжении всего жизненного цикла. Это обеспечивает более высокое качество данных, так как только актуальная и необходимая информация будет зарегистрирована в системе управления метаданными.

7) Слой оркестрации и мониторинга

Компоненты: Kubernetes, Prometheus.

Роль и функции. Оркестрация и мониторинг состояния системы.

Причины выбора. Prometheus изначально разработан с учетом работы в среде Kubernetes. Он автоматически обнаруживает новые сервисы, что позволяет легко настраивать мониторинг без необходимости вручную добавлять каждый новый компонент. Kubernetes позволяет легко масштабировать приложения, добавляя или удаляя точки соединения в зависимости от нагрузки. Prometheus может адаптироваться к этим изменениям, автоматически обнаруживая новые ресурсы и начиная собирать с них метрики.

2.2. Проектирование архитектуры

- Разработать схему потоков данных.

Сбор данных

Данные о состоянии сети собираются с помощью SNMP Collector, который отправляет запросы к SNMP-агентам на управляемых устройствах.

Эти данные с помощью Zabbix передаются в Apache Kafka для обработки потоков в реальном времени.

Передача и хранение

Kafka передает собранные данные в Greenplum для структурированных и полуструктурированных данных, а также в Cassandra для неструктурированных и полуструктурированных данных.

Обработка и анализ

Данные обрабатываются с использованием Apache Spark и Flink, что позволяет выполнять сложные вычисления и анализировать данные в режиме реального времени.

Результаты анализа передаются в TensorFlow для создания моделей машинного обучения и визуализируются с помощью PowerBI и PyCharm.

Управление данными

Apache Atlas управляет метаданными, а Apache NiFi автоматизирует потоки данных, обеспечивая их правильное направление и обработку.

Мониторинг и оркестрация

Kubernetes управляет контейнерами, а Prometheus отслеживают состояние всей системы, обеспечивая высокую доступность и производительность.

2.3 Определить компоненты для обеспечения безопасности и управления доступом.

Для обеспечения безопасности и управления доступом можно использовать следующие компоненты:

Многофакторная аутентификация для повышения уровня безопасности при доступе к критическим ресурсам.

Реализация контроля доступа к данным на уровне источников, что позволяет ограничивать доступ в зависимости от ролей пользователей.

Каждое обращение к ресурсам должно требовать аутентификации, независимо от местоположения пользователя.

2.4 Схема хранения (Звезда или снежинка)

Аргументы в пользу схемы «Снежинка».

С технической точки зрения:

1) Большой объем данных: 1 ПБ данных в год с ростом 40% ежегодно - это огромный объем информации. Схема «снежинка» позволяет эффективно хранить и управлять такими данными за счет нормализации.

2) Разнообразие типов данных. Схема «снежинка» позволяет хранить все типы данных, обеспечивая при этом оптимальную структуру для анализа.

3) Высокая скорость получения данных: «снежинка» с нормализованными данными позволяет эффективно обрабатывать большие объемы информации, не создавая избыточных запросов к хранилищу.

4) Анализ качества связи и прогнозирование нагрузки на сеть: «снежинка» позволяет создавать агрегированные таблицы, которые упрощают анализ данных и прогнозирование нагрузки на сеть. Агрегированные данные могут храниться отдельно от исходных.

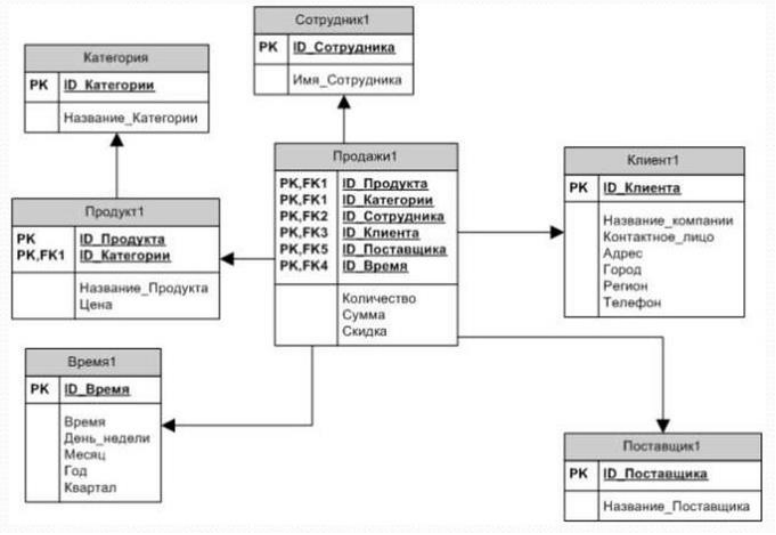
С экономической точки зрения:

1) Эффективное использование дискового пространства. Нормализация данных в схеме «снежинка» позволяет сократить дублирование информации, что снижает потребность в дисковом пространстве и, соответственно, затраты на хранение данных.

2) Упрощение администрирования. «Снежинка» позволяет эффективно управлять данными, упрощая администрирование системы хранения данных.

3) Оптимизированная структура данных в схеме «снежинка» позволяет выполнять запросы к данным быстрее, что повышает производительность аналитических систем.

Схема «снежинка» является более подходящей для телекоммуникационной компании, учитывая масштабы данных, разнообразие типов данных и требования к обработке. Она обеспечивает эффективное хранение данных, оптимизирует использование дискового пространства, упрощает администрирование и повышает производительность аналитических систем. Пример такой схемы:



Для телекоммуникационной компании основной таблицей будет количество оказываемых услуг по их типу и по пользователям с детализацией на маркетинговые компании для каждого типа услуги и ее подкатегории.

2.5. Lambda-архитектура или Kappa-архитектура

Для телекоммуникационной компании с объемом данных 1 ПБ в год и ростом 40% ежегодно Lambda-архитектура будет более подходящим выбором

Причины выбора:

1. Обработка данных в реальном времени и пакетная обработка

Lambda-архитектура сочетает в себе два подхода: потоковую обработку (speed layer) для анализа данных в реальном времени и пакетную обработку (batch layer) для анализа исторических данных. Это позволит компании одновременно обрабатывать текущие события, хранить и анализировать большие объемы исторических данных.

Kappa-архитектура, напротив, фокусируется исключительно на потоковой обработке.

2. Гибкость и отказоустойчивость

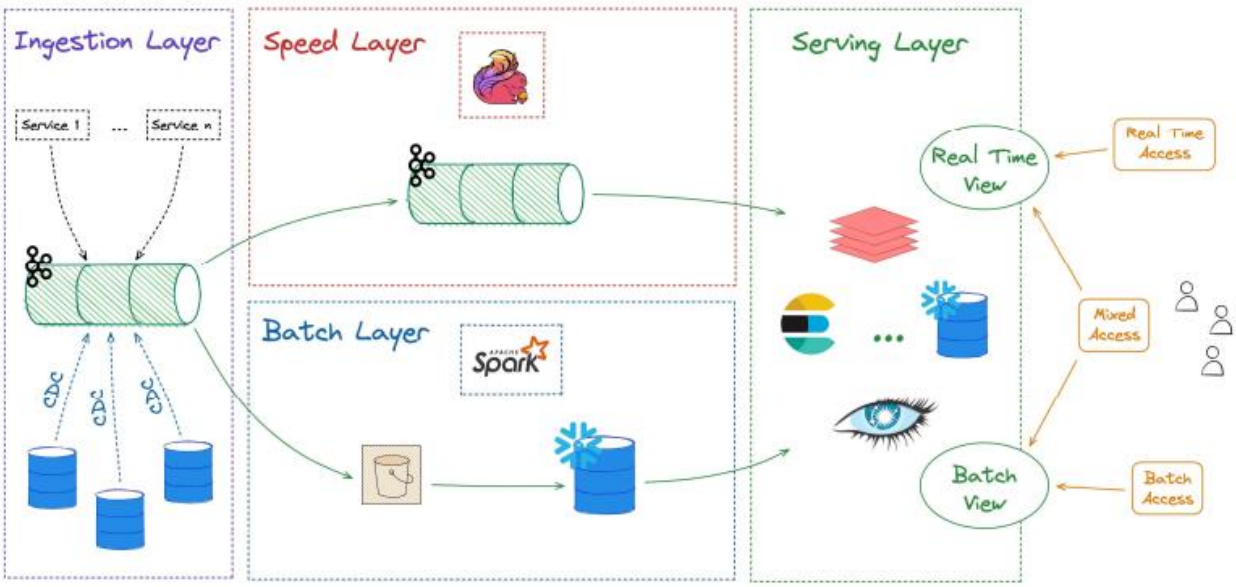
Lambda-архитектура предоставляет возможность использовать разные технологии для потоковой и пакетной обработки, если одна часть системы выходит из строя, другая может продолжать работу.

Kappa-архитектура требует полной зависимости от потоковой обработки.

3. Поддержка сложных аналитических запросов

Lambda-архитектура позволяет выполнять сложные OLAP-запросы на исторических данных, что критично для анализа качества связи и прогнозирования нагрузки.

Kappa-архитектура менее оптимизирована для выполнения таких запросов, так как ориентирована на простую потоковую обработку.



2.6. OLAP vs OLTP

Традиционно телекоммуникационные и другие организации, занятые в сфере услуг, собирают и хранят в своих базах терабайты данных о клиентах, продуктах и сервисах. Для подобного бизнеса используют системы оперативной обработки данных OLTP.

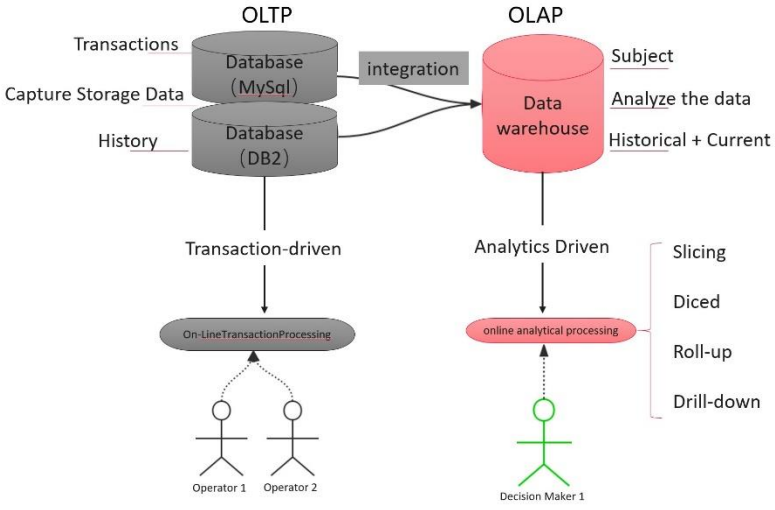
- Используется для обработки транзакций в реальном времени.
- Подходит для систем, которые требуют высокой скорости обработки записей (например, работа с базами данных клиентов, оплаты, операции с тарифами).
- Обеспечивает быструю запись и обновление данных.

В то же время по условиям задачи указаны требования к обработке данных: анализ качества связи, прогнозирование нагрузки на сеть. Для этого больше подходит OLAP.

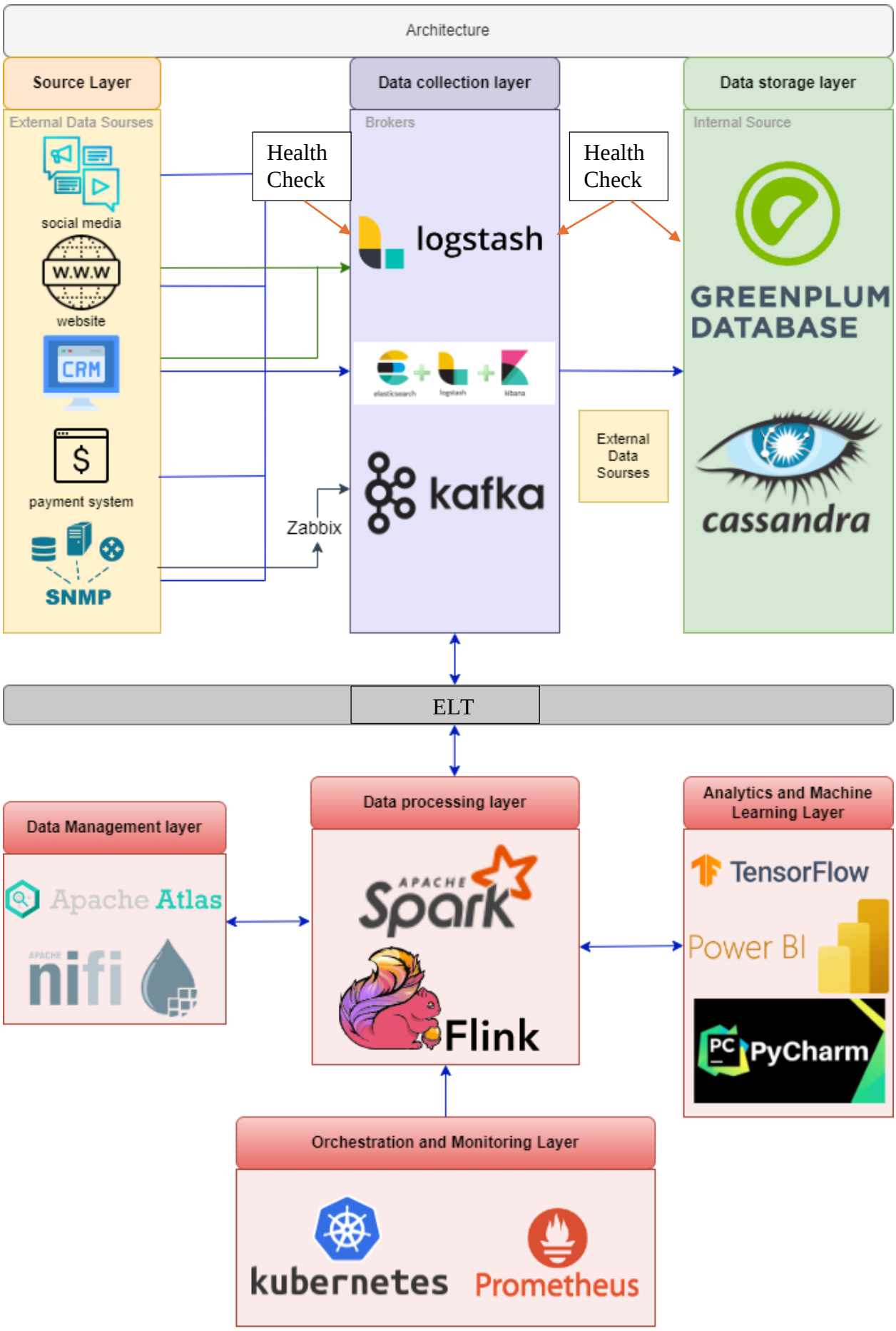
- Ориентирован на анализ и обработку больших объёмов данных.
- Подходит для создания отчётов, анализа производительности и принятия стратегических решений.
- Поддерживает сложные запросы к данным и агрегацию.

Для телекоммуникационной компании при данных условиях будет полезно использовать обе технологии в разных областях:

- OLTP для управления повседневными операциями, взаимодействия с клиентами и обработки вновь поступающих данных.
- OLAP для анализа данных, создания отчётов и принятия решений на основе исторической информации и исторических сводок.



2.7. Создание диаграммы архитектуры:


Architecture
Source Layer
External Data Sourses
social media
W.W.W
website
CRM
payment system
SNMP
Data collection layer
Brokers
Health Check
Health Check
logstash
kafka
Zabbix
Data storage layer
Internal Source
GREENPLUM DATABASE
cassandra
External Data Sourses
ELT
Data Management layer
Apache Atlas
nifi
Data processing layer
Spark
Flink
Analytics and Machine Learning Layer
TensorFlow
Power BI
PyCharm
Orchestration and Monitoring Layer
kubernetes
Prometheus

3. Рассмотрение вопросов производительности и масштабируемости:

3.1 Описать, как архитектура обеспечивает высокую производительность и масштабируемость.

Стратегия масштабирования должна учитывать следующие аспекты:

Горизонтальное масштабирование. Добавление новых узлов в кластер Greenplum или Cassandra по мере роста объема данных.

Использование Kubernetes для автоматического добавления ресурсов в зависимости от нагрузки.

Внедрение Redis или Memcached для кэширования часто запрашиваемых данных, что снизит нагрузку на базу данных.

Разделение нагрузки посредством балансировщиков нагрузки для распределения запросов между несколькими экземплярами приложений.

3.2 Анализ потенциальных проблем и их решений:

- Выявить возможные узкие места в архитектуре.

1) Задержки при передаче данных. Если SNMP Collector или другие компоненты сбора данных не справляются с высокой нагрузкой (до 10,000 событий в секунду), это может привести к задержкам в передаче данных в брокер (например, Apache Kafka).

2) Производительность базы данных. Greenplum может испытывать трудности с производительностью при выполнении сложных аналитических запросов, особенно при увеличении объема данных (рост 40% ежегодно).

3) Масштабируемость Cassandra. При увеличении объема неструктурированных данных могут возникнуть проблемы с производительностью и доступностью, если система не настроена должным образом.

4) Управление метаданными. Без эффективного управления метаданными может возникнуть путаница в структуре данных и их источниках, что затруднит анализ и отчетность.

- Предложить способы их устранения или минимизации влияния

1) Оптимизация SNMP Collector. Необходимо использовать более эффективные алгоритмы сбора данных и распределять нагрузку между несколькими экземплярами Collectors.

2) Настроить кластеризацию Greenplum для распределения нагрузки на несколько узлов базы данных. Это позволит улучшить производительность при выполнении аналитических запросов.

3) Регулярно анализировать метрики производительности и масштабировать систему по мере необходимости. Настройка автоматического масштабирования может помочь справиться с увеличением нагрузки.

4) Разработать API для упрощения интеграции различных компонентов системы, что позволит избежать задержек и несогласованности данных.

5) Регулярное тестирование и мониторинг системы с использованием Prometheus для отслеживания состояния всех компонентов архитектуры в реальном времени, что позволит быстро реагировать на возникающие проблемы.

Вывод: в ходе проделанной работы была построена архитектура хранилища данных, аргументирован выбор каждого компонента, а также была описано, как архитектура обеспечивает высокую производительность и масштабируемость, был проделан анализ потенциальных проблем и их решений.