МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра математического моделирования и анализа данных

РЕФЕРАТ

РАЗЛИЧНЫЕ ПОДХОДЫ К ХРАНЕНИЮ БОЛЬШИХ ДАННЫХ

Подготовили:

Бовт Тимофей Анатольевич 4 курс 7 группа

Гут Валерия Александровна 4 курс 7 группа

Руководитель:

Сталевская Светлана Николаевна

Минск

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Big Data — это структурированные, частично структурированные или неструктурированные большие массивы данных. Также под этим термином понимают обработку, хранение и анализ огромных объемов данных. То есть, когда у вас так много информации, что обычные методы работы с ней становятся неэффективными.

Примеры больших данных:

- информация о лайках, комментариях, активности пользователей в социальных сетях за месяц;
- массив данных о населении в городской системе распознавания лиц;
- логи серверов, где каждый день генерируются миллиарды записей;
- медицинские записи и образы, включая результаты анализов за десятилетия в Единой медицинской информационно-аналитической системе;
- аналитика поисковых систем, собирающая запросы от миллиардов пользователей.

Если говорить простыми словами, большие данные — более крупные и сложные наборы данных, особенно из новых источников данных. Размер этих наборов данных настолько велик, что традиционные программы для обработки не могут с ними справиться. Однако эти большие данные можно использовать для решения бизнес-задач, которые раньше не могли быть решены.

Границы термина Big Data размыты и могут отличаться в зависимости от конкретной задачи. Тем не менее существуют три основных характеристики Big Data:

Volume. Количество данных — важный фактор. Располагая ими в больших количествах, Вам потребуется обрабатывать большие объемы неструктурированных данных низкой плотности. Ценность таких данных не всегда известна. Это могут быть данные каналов Twitter, данные посещаемости веб-страниц, а также данные мобильных приложений, сетевой трафик, данные датчиков. В некоторые организации могут поступать десятки терабайт данных, в другие — сотни петабайт.

Velocity. Скорость в данном контексте — это скорость приема данных и, возможно, действий на их основе. Обычно высокоскоростные потоки данных поступают прямо в оперативную память, а не записываются на диск. Некоторые "умные" продукты, функционирующие на основе Интернета, работают в

режиме реального или практически реального времени. Соответственно, такие данные требуют оценки и действий в реальном времени.

Variety. Разнообразие означает, что доступные данные принадлежат к разным типам. Традиционные типы данных структурированы и могут быть сразу сохранены в реляционной базе данных. С появлением Big Data данные стали поступать в неструктурированном виде. Такие неструктурированные и полуструктурированные типы данных как текст, аудио и видео, требуют дополнительной обработки для определения их значения и поддержки метаданных.

В дальнейшем появилась интерпретация с «пятью V»:

Viability. Жизнеспособность данных. При большом разнообразии данных и переменных, необходимо проверять их значимость при построении модели прогнозирования.

Value. Ценность данных. После подтверждения жизнеспособности специалисты Big Data изучают взаимосвязи данных. После оценки дополнительных переменных прогнозная модель становится сложнее и эффективнее.

ГЛАВА 1. АНАЛИТИКА БОЛЬШИХ ДАННЫХ

Существует четыре типа аналитики Big Data: описательная, диагностическая, предиктивная и предписывающая.

Описательная аналитика позволяет узнать, что и когда произошло. Например, с ее помощью компании могут определить, какой продукт или канал работает лучше всего.

Диагностический анализ показывает, почему произошло то или иное событие. Для работы с информацией используется интеллектуальный анализ данных (Data Mining). Он позволяет извлекать данные из массива, находить у них определенные закономерности, тренды или отклонения и классифицировать по схожим признакам.

С помощью диагностического анализа компания может, например, выявить причины падения продаж или оттока клиентов.

Предиктивная аналитика позволяет спрогнозировать возможности и оценить риски для более точных и эффективных бизнес-решений. Здесь для работы с данными активно используются искусственный интеллект и машинное обучение — ML (machine learning).

Хороший пример использования ML в предиктивной аналитике — кредитный скоринг в банках. Если раньше всю аналитическую работу по оценке рисков невозврата кредита выполняли сотрудники банков, то с внедрением ML завяки на кредит стали обрабатываться автоматически. Теперь сотруднику не нужно изучать текст заявки и сравнить необходимые метрики с какими-то профилями должников. За него это делает модель, обученная на тысячах кредитных заявок.

Предписывающая аналитика дает рекомендации о том, что следует сделать и как оптимизировать процессы. Один из примеров — ретейлеры с помощью такой аналитики оптимизируют ассортимент товаров и цены на них с учетом модели поведения покупателей.

Указанные типы также можно встретить в контексте **модели зрелости аналитики**. Она показывает, на каком этапе эволюции находится компания согласно своей способности управлять большими (или стандартными) данными и извлекать из них пользу.

Работа с Big Data состоит из нескольких этапов. Сначала данные собирают в хранилище, очищают и лишь затем анализируют. Рассмотрим каждый этап подробнее.

Любая работа с данными — Big Data и не только — начинается с того, что их необходимо получить. Для этого используются различные источники, от социальных сетей до веб-сайтов.

Главные источники для генерируемых людьми социальных данных — соцсети, веб, GPS-данные о перемещениях. Также специалисты Big Data используют данные, ложащиеся в основу статистических показателей: сырые данные переписей населения, налоговых взносов, документации, по которой определяется рождаемость и смертность.

Транзакционная информация появляется при любых денежных операциях и взаимодействии с банкоматами: переводах, покупках, поставках.

Источником машинных данных служат смартфоны, IoT-гаджеты, автомобили и другая техника, датчики, системы слежения и спутники.

Существует два подхода к извлечению данных.

- **Полное извлечение**, при котором нет потребности отслеживать изменения. Процесс проще, но нагрузка на систему выше.
- **Инкрементное извлечение**. Изменения в исходных данных отслеживают с момента последнего успешного извлечения. Для этого создают таблицы изменений или проверяют временные метки. Многие хранилища имеют встроенную функцию захвата данных об изменениях (CDC), которая позволяет сохранить состояния данных. Логика для инкрементального извлечения более сложная, но нагрузка на систему снижается.

Поступившие данные необходимо где-то хранить — для этого применяются хранилища данных (Data Warehouse) и озера данных (Data Lake).

Data Warehouse использует принцип ETL (Extract, Transform, Load) — сначала идет извлечение, далее преобразование, потом загрузка. Здесь находятся структурированные и уже обработанные данные и используются реляционные или аналитические СУБД Greenplum, ClickHouse, Apache Doris.

Data Lake отличается методом ELT (Extract, Load, Transform) — сначала загрузка, следом преобразование данных. Здесь находятся все нужные данные в

необработанном виде: и структурированные, и неструктурированные. Обычно это Hadoop (HDFS), либо объектные хранилища (MinIO, S3 или любое другое облачное хранилище).

Существуют три принципа хранения Big Data.

- **Горизонтальное масштабирование**. Система должна иметь возможность расширяться. Если объем данных вырос, необходимо увеличить мощность кластера путем добавления серверов.
- **Отказоустойчивость**. Для обработки требуются большие вычислительные мощности, что повышает вероятность сбоев. Большие данные должны обрабатываться непрерывно в режиме реального времени.
- **Локальность**. В кластерах применяется принцип локальности данных — обработка и хранение происходит либо в рамках одного кластера, либо при обеспечении максимально быстрой скорости передачи информации. Такой подход минимизирует расходы мощностей на передачу информации между серверами.

При работе с большими данными требуется Data Cleaning — выявление, очистка и исправление ошибок, нерелевантной информации и несоответствий данных. Процесс позволяет оценить косвенные показатели, погрешности, пропущенные значения и отклонения. Как правило, во время извлечения данные преобразуются. Специалисты Big Data добавляют дополнительные метаданные, временные метки или геолокационные данные.

На этапе анализа больших данных (Big Data analysis) очищенные данные анализируют, а полученные результаты интерпретируют. При этом анализ выполняется постоянно, в режиме реального времени.

Для анализа и оценки данных существует множество различных инструментов — выбор подходящего зависит от ваших целей, задач и личных предпочтений.

ГЛАВА 2. ИНСТРУМЕНТЫ ДЛЯ РАБОТЫ С БОЛЬШИМИ ДАННЫМИ

Hadoop – это свободно распространяемый набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов. Эта основополагающая технология хранения и обработки больших данных (Big Data) является проектом верхнего уровня фонда Apache Software Foundation.

Изначально проект разработан на Java в рамках вычислительной парадигмы MapReduce, когда приложение разделяется на большое количество одинаковых элементарных заданий, которые выполняются на распределенных компьютерах (узлах) кластера и сводятся в единый результат.

Проект состоит из основных 4-х модулей:

1. **Hadoop Common** – набор инфраструктурных программных библиотек и утилит, которые используются в других решениях и родственных проектах, в частности, для управления распределенными файлами и создания необходимой инфраструктуры;
2. **HDFS** – распределённая файловая система, Hadoop Distributed File System – технология хранения файлов на различных серверах данных (узлах, DataNodes), адреса которых находятся на специальном сервере имен (мастере, NameNode)**.** За счет дублирования (репликации) информационных блоков, HDFS обеспечивает надежное хранение файлов больших размеров, поблочно распределённых между узлами вычислительного кластера;
3. **YARN** – система планирования заданий и управления кластером (Yet Another Resource Negotiator), которую также называют MapReduce 2.0 (MRv2) – набор системных программ (демонов), обеспечивающих совместное использование, масштабирование и надежность работы распределенных приложений**.** Фактически, YARN является интерфейсом между аппаратными ресурсами кластера и приложениями, использующих его мощности для вычислений и обработки данных;
4. **Hadoop MapReduce** – платформа программирования и выполнения распределённых MapReduce-вычислений, с использованием большого количества компьютеров (узлов, nodes), образующих кластер.

Apache Spark – это Big Data фреймворк с открытым исходным кодом для распределённой пакетной и потоковой обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop. Спарк состоит из следующих **компонентов**:

- **Ядро** (Core);
- **SQL** – инструмент для аналитической обработки данных с помощью SQL-запросов;
- **Streaming** – надстройка для обработки потоковых данных, о которой подробно мы рассказывали здесь и здесь;
- **MLlib** – набор библиотек машинного обучения;
- **GraphX** – модуль распределённой обработки графов.

Spark может работать как в среде кластера Hadoop под управлением YARN, так и без компонентов ядра хадуп, например, на базе системы управления

кластером Mesos. Спарк поддерживает несколько популярных распределённых систем хранения данных (HDFS, OpenStack Swift, Cassandra, Amazon S3) и языков программирования (Java, Scala, Python, R), предоставляя для них API-интерфейсы.

Справедливости ради стоит отметить, что Spark Streaming, в отличие от, например, Apache Storm, Flink или Samza, не обрабатывает потоки Big Data целиком. Вместо этого реализуется микропакетный подход (micro-batch), когда поток данных разбивается на небольшие пакеты временны*х* интервалов. Абстракция Spark для потока называется DStream (discretized stream, дискретизированный поток) и представляет собой микро-пакет, содержащий несколько отказоустойчивых распределенных датасетов, **RDD** (resilient distributed dataset).

Именно RDD является основным вычислительным примитивом Спарк, над которым можно делать параллельные вычисления и преобразования с помощью встроенных и произвольных функций, в том числе с помощью временных окон (window-based operations).

Благодаря наличию разнопрофильных инструментов для аналитической обработки данных «на лету» (SQL, Streaming, MLLib, GraphX), Спарк активно используется в системах интернета вещей (Internet of Things, IoT) на стороне IoT-платформ, а также в различных бизнес-приложениях, в т.ч. на базе методов Machine Learning. Например, Спарк применяется для прогнозирования оттока клиентов (Churn Predict) и оценки финансовых рисков. Однако, если временная задержка обработки данных (latency) – это критичный фактор, Apache Spark не подойдет и стоит рассмотреть альтернативу в виде клиентской библиотеки Kafka Streams или фреймворков Storm, Flink, Samza.

NoSQL — тип нереляционных СУБД. Хранение и поиск данных моделируется отличными от табличных отношений средствами. Для хранения информации не требуется заранее заданная схема данных. Главное преимущество подобного подхода — любые данные можно быстро помещать и извлекать из хранилища. Термин расшифровывается как Not Only SQL.

Примеры таких СУБД:

Cassandra — база данных от Apache, хорошо масштабируется и позволяет обрабатывать Big Data с помощью репликации на несколько узлов;
MongoDB — расширяемая документоориентированная СУБД для разработчиков с открытым исходным кодом, распространяется под GNU AGPL.

Также на просторах интернета можно встретить информацию о методе параллельной обработки MapReduce, разработанном компанией Google. Этот подход теряет популярность, но знать о нем — полезно.

MapReduce организовывает данные в виде записей. Функции здесь работают независимо и параллельно, что обеспечивает соблюдение принципа горизонтальной масштабируемости. Обработка происходит в три стадии.

Map. Функцию определяет пользователь, map служит для начальной обработки и фильтрации. Функция применима к одной входной записи, она выдает множество пар ключ-значение. Применяется на том же сервере, на котором хранятся данные, что соответствует принципу локальности.

Shuffle. Вывод map разбирается по «корзинам». Каждая соответствует одному ключу вывода первой стадии, происходит параллельная сортировка. «Корзины» служат входом для третьей стадии.

Reduce. Каждая «корзина» со значениями попадает на вход функции reduce. Ее задает пользователь и вычисляет финальный результат для каждой «корзины». Множество всех значений функции reduce становится результатом.

Самые популярные языки программирования для работы с большими данными

Scala. Нативный язык для Apache Spark, используется для анализа данных. Проекты Apache Software Foundation, Spark и Kafka, написаны в основном на Scala.

Python. Обладает готовыми библиотеками для работы с AI, ML и другими методами статистических вычислений: TensorFlow, PyTorch, SKlearn, Matplotlib, Scipy, Pandas. Для обработки и хранения данных существуют API в большинстве фреймворков: Apache Kafka, Spark, Hadoop.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://www.oracle.com/cis/big-data/what-is-big-data/#challenges
2. https://selectel.ru/blog/what-is-big-data/
3. https://bigdataschool.ru/wiki/spark
4. https://bigdataschool.ru/wiki/hadoop