Министерство науки и высшего образования РФ
ФГБОУ ВО «Ульяновский государственный университет»
Факультет математики, информационных и авиационных технологий

Кафедра телекоммуникационных технологий и сетей

Липатова Светлана Валерьевна

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
для лабораторного практикума
и самостоятельной работы
по дисциплине

«Технологии обработки больших данных»

для студентов направления
11.04.02 " Инфокоммуникационные технологии и системы связи"

Ульяновск
2022

Методические рекомендации для семинарских (практических) занятий, лабораторного практикума и самостоятельной работы по дисциплине «Технологии обработки больших данных» / составитель: С.В. Липатова - Ульяновск: УлГУ, 2022 –29 с.

Настоящие методические рекомендации предназначены для студентов направления обучения 11.04.02 "Инфокоммуникационные технологии и системы связи". В работе приведены материалы для самостоятельного изучения и контроля усвоения материала.

Студентам всех форм обучения следует использовать данные методические рекомендации при подготовке к семинарам, самостоятельной подготовке, а также промежуточной аттестации по дисциплине «Технологии обработки больших данных».

Рекомендованы к введению в образовательный процесс

Учёным советом факультета математики, информационных и авиационных технологий УлГУ

протокол № 3/19 от «19» апреля 2022 г.

СОДЕРЖАНИЕ

ОБЩИЕ ВОПРОСЫ

В результате изучения дисциплин «Технологии обработки больших данных» студенты должны:

- приобретение студентами знаний о технологиях подготовки, хранения, обработки и анализа больших данных;
- применение статистических и математических методов для анализа больших объёмов информации;
- приобретение практических навыков работы с нереляционными базами данных,
- приобретение студентами знаний о экосистеме Hadoop.

Методические рекомендации для семинарских (практических) занятий, лабораторного практикума и самостоятельной работы по дисциплине «Технологии обработки больших данных» направлены на повышение эффективности освоения знаний, умений, навыков и компетенций, связанных с использованием библиотек на языке Python для решения задач машинного обучения.

Методические рекомендации предлагают указания по всем темам дисциплины «Технологии обработки больших данных». Методические рекомендации разбиты по темам и содержат набор вопросов для систематизации теоретического материала, полученного на лекционных занятиях, и самостоятельного изучения теории, вопросы (тесты) для текущего контроля на практических занятиях (семинарах), задачи для усвоения практических навыков. Для лабораторного практикума приведены задания, варианты и рекомендации по выполнению лабораторных работ.

Список литературы и информационного обеспечения, приведённый в конце методических указаний, может служить основой для изучения всех рассматриваемых тем. Дополнительная и учебно-методическая литература могут быть использованы обучающимися для закрепления изучаемого материала.

РЕКОМЕНДАЦИИ ПО ТЕМАМ ДИСЦИПЛИНЫ

Тема 1. Введение в BigData.

Основные вопросы темы

Основные вызовы больших данных(V). Определение термина "большие данные". Классификация Big Data.Роли игроков на рынке Big Data. Профессии Big Data. Примеры успешных кейсов.

Материалы для самоподготовки

1. **Что такое большие данные?**

Это серия подходов, инструментов и методов обработки структурированных и неструктурированных данных огромных объёмов и значительного многообразия для получения воспринимаемых человеком результатов, эффективных в условиях непрерывного прироста и распределения по многочисленным узлам вычислительной сети, альтернативных традиционным системам управления базами данных.

2. **Перечислите основные V для больших данных?**

- Volume – объем, накопленная база данных представляет собой большой объем информации, который трудоемко обрабатывать и хранить традиционными способами, для них требуются новый подход и усовершенствованные инструменты.
- Velocity – скорость, данный признак указывает как на увеличивающуюся скорость накопления данных (90% информации было собрано за последние 2 года), так и на скорость обработки данных, в последнее время стали более востребованы технологии обработки данных в реальном времени.
- Variety – многообразие, т.е. возможность одновременной обработки структурированной и неструктурированной разно- форматной информации. Главное отличие структурированной информации – это то, что она может быть классифицирована. Примером такой информации может служить информация о клиентских транзакциях. Неструктурированная информация включает в себя видео, аудио файлы, свободный текст, информацию, поступающую из социальных сетей. На сегодняшний день 80% информации входит в группу неструктурированной. Данная информация нуждается в комплексном анализе, чтобы сделать ее полезной для дальнейшей обработки.
- Veracity – достоверность данных, все большее значение пользователи стали придавать значимость достоверности имеющихся данных. Так, у интернет-

компаний есть проблема по разделению действий, проводимых роботом и человеком на сайте компании, что приводит в конечном счете к затруднению анализа данных.

- Value – ценность накопленной информации. Большие Данные должны быть полезны компании и приносить определенную ценность для нее. К примеру, помогать в усовершенствовании бизнес- процессов, составлении отчетности или оптимизации расходов.

3. Как можно классифицировать большие данные?

Дайон Хинчклиф, редактора журнала Web 2.0 Journal делит Большие данные на 3 группы:

- Быстрые Данные (Fast Data), их объем измеряется терабайтами;
- Большая Аналитика (Big Analytics) — петабайтные данные
- Глубокое Проникновение (Deep Insight) — экзабайты, зеттабайты.

4. Основные профессии в больших данных?

- исследователь данных
- консультант в области больших данных
- инженер по большим данным
- архитектор больших данных
- специалист по управлению большими данными

5. Игроки на рынке больших данных?

- **Поставщики инфраструктуры** — решают задачи хранения и предобработки данных.

Например: IBM, Microsoft, Oracle, Sap и другие.

- **Датамайнеры —** разработчики алгоритмов, которые помогают заказчикам извлекать ценные сведения.

Среди них: Yandex Data Factory, «Алгомост», Glowbyte Consulting, CleverData и др.

- **Системные интеграторы** — компании, которые внедряют системы анализа больших данных на стороне клиента.

К примеру: «Форс», «Крок» и др.

- **Потребители** — компании, которые покупают программно-аппаратные комплексы и заказывают алгоритмы у консультантов.

Это «Сбербанк», «Газпром», «МТС», «Мегафон» и другие компании из отраслей финансов, телекоммуникаций, ритейла.

– **Разработчики готовых сервисов** — предлагают готовые решения на основе доступа к большим данным. Они открывают возможности Big Data для широкого круга пользователей.

6. **Направления больших данных?**

– Сбор и обработка больших данных
– Аналитика
– Инженерия больших данных
– Архитектура больших данных и системная интеграция
– Разработка продуктов и услуг на основе больших данных
– Управление большими данными и системами на основе больших данных
– Проведение исследований с целью получения новых математических и технических решений для работы с большими данными

Тема 2. Хранилища данных. Технологии OLTP, OLAP, ETL.

Основные вопросы темы

Принципы технологии OLTP. Понятие транзакции. Способы организации транзакций и принципы блокировки доступа к данным. Назначение технологии.

Определение и свойства хранилищ данных, виды данных, хранящихся в хранилищах. Многомерная модель представления данных. Технологии BI и ETL, OLAP. Виды реализации многомерной модели данных. СУБД, обеспечивающие поддержку OLAP.

Материалы для самоподготовки

1. **Что такое хранилище данных?**

Это предметно-ориентированное, привязанное ко времени и неизменяемое собрание данных для поддержки процесса принятия управляющих решений

2. **Свойства хранилищ?**

- данные предметно-ориентированны (учитывается специфика предметной области, информация в ХД предназначена для решения задачи поддержки принятия решений, т.е. присутствуют "исторические" данные - факты за определенные интервалы времени, структуры данных отражают развитие всех направлений бизнес-процесса компании во времени);
- интегрированы и внутренне непротиворечивы (при поступлении из разнородных источников оперативной информации должны быть обеспечены, очистка и

согласованность данных для формирования единого информационного пространства);

- данные инвариантны во времени (данные сохраняют свою истинность в любой момент процесса чтения, в оперативном режиме они не обновляются, а лишь регулярно пополняются из систем оперативной обработки по заданной дисциплине);
- поддерживающие хронологию (однажды загруженные данные теоретически никогда не меняются, по отношению к ним возможны только две операции: начальная загрузка и чтение);
- полнота и достоверность хранимых данных(наборы данных, организованные с целью поддержки управления, призванные выступать в роли единого и единственного источника информации, обеспечивающего менеджеров и аналитиков достоверной информацией, необходимой для оперативного анализа и поддержки принятия решений).

3. OLTP (ONLINE TRANSACTION PROCESSING)?

- **транзакционная система** — обработка транзакций в реальном времени. Способ организации БД, при котором система работает с небольшими по размерам транзакциями, но идущими большим потоком, и при этом клиенту требуется от системы минимальное время отклика.
- Термин OLTP применяют также к системам (приложениям). OLTP-системы предназначены для ввода, структурированного хранения и обработки информации (операций, документов) в режиме реального времени.

4. OLAP (ONLINE ANALYTICAL PROCESSING)?

- технология обработки данных, заключающаяся в подготовке суммарной (агрегированной) информации на основе больших массивов данных, структурированных по многомерному принципу.

5. 12 признаков OLAP-данных по Кодду?

1) Многомерная концепция данных. OLAP оперирует CUBE-данными, которые являются многомерными массивами. Число измерений OLAP-кубов не ограничено.
2) Прозрачность. OLAP-системы должны опираться на открытые системы, поддерживающие гетерогенные источники данных.
3) Доступность. OLAP-системы должны представлять пользователю единую логическую схему данных.
4) Постоянная скорость выполнения запросов. Производительность не должна падать при росте числа измерений.

5) Клиент/сервер архитектура. Системы должны базироваться на открытых интерфейсах и иметь модульную структуру.
6) Различное число измерений. Системы не должны ограничиваться трехмерной моделью представления данных. Измерения должны быть эквивалентны по применению любых функций.
7) Динамическое представление разреженных матриц. Под разреженной матрицей понимается матрица, не каждая ячейка которой содержит данные. OLAP- системы должны содержать средства хранении и обработки разреженных матриц больших объёмов.
8) Многопользовательская поддержка. OLAP-системы должны поддерживать многопользовательский режим работы.
9) Неограниченные многомерные операции. Аналогично требованию о различном числе измерений: все измерения считаются равными, и многомерные операции не должны накладывать ограничения на отношения между ячейками.
10) Интуитивно понятные инструменты манипулирования данными. Для формулирования многомерными запросами пользователи не должны работать в усложнённых меню.
11) Гибкая настройка конечных отчётов. Пользователи должны иметь возможность видеть только необходимые им данные, причём все их изменения должны немедленно отображаться в отчётах.
12) Отсутствие ограничений. Отсутствие ограничений на количество измерений и уровней агрегации данных.

6. Тест FASMI (Fast Shared Multidimensional Information)?

- Fast – быстрой, обеспечивать почти мгновенный отклик на большинство запросов;
- Shared – многопользовательской, должен существовать механизм контроля доступа к данным и возможность одновременной работы многих пользователей;
- Multidimensional – многомерной. Данные должны представляться в виде многомерных кубов;
- Information – данные должны быть полны с точки зрения аналитика, т.е. содержать всю необходимую информацию.

7. Реализация OLAP?

Физическая OLAP. Программа, выполняющая на этапе предварительной загрузки данных в OLAP предварительный расчёт агрегатов, которые затем сохраняются в специальную многомерную базу данных, обеспечивающую быстрое извлечение и экономичное хранение.

Виртуальная OLAP. Все данные хранятся и обрабатываются в реляционных системах управления базами данных, а агрегаты могут не существовать вообще или создаваться по первому запросу в СУБД или кэше аналитического программного обеспечения.

Гибридная OLAP. Реализация является комбинацией: сами данные хранятся в реляционной базе данных, а агрегаты — в многомерной.

Тема 3. Технологии NoSQL.

Основные вопросы темы

Горизонтальное и вертикальное масштабирование. CAP-теорема. История термина NoSQL и его трактование. BASE–архитектура (Basically Available, Soft-state, Eventually consistent). Графовые, колончатые, докменто-ориентированные модели модель и ключ-значение. Термин NewSQL. СУБД: HBase, Cassandra, Neo4j, MongoDb.

Материалы для самоподготовки

1. **Что значит NoSQL?**

NoSQL – термин расшифровывается как Not Only SQL (не только SQL). Включает в себя ряд подходов, направленных на реализацию базы данных, имеющих отличия от моделей, используемых в традиционных, реляционных СУБД. Их удобно использовать при постоянно меняющейся структуре данных. Например, для сбора и хранения информации в социальных сетях.

2. **Какие недостатки у реляционных БД?**

– «дисперсионное хранение»: данные об объекте в разных таблицах
– производительность падает при сложных запросах
– плохо приспособлены для работе в кластере или облаке
– хорошо работают со структурированными данными, структура которых редко меняется, для других данных мало приспособлены

3. **BASE–архитектура (Basically Available, Soft-state, Eventually consistent)?**

Базовая доступность *(*Basically Available*)*: сбой в некоторых узлах приводил к отказу в обслуживании только для незначительной части сессий при сохранении доступности в большинстве случаев

Неустойчивое состояние *(*Soft-state*):* возможность жертвовать долговременным хранением состояния сессий (таких как промежуточные результаты выборок, информация о навигации, контексте), при этом концентрируясь на фиксации обновлений только критичных операций.

Согласованности в конечном счёте *(*Eventually consistent*):* возможность противоречивости данных в некоторых случаях, но при обеспечении согласования в практически обозримое время.

4. **CAP –теорема (теорема Брюера)?**

в любой реализации распределённых вычислений возможно обеспечить не более двух из трёх следующих свойств:

согласованность данных (англ. consistency) — во всех вычислительных узлах в один момент времени данные не противоречат друг другу;

доступность (англ. availability) — любой запрос к распределённой системе завершается корректным откликом, однако без гарантии, что ответы всех узлов системы совпадают;

устойчивость к разделению (англ. partition tolerance) — расщепление распределённой системы на несколько изолированных секций не приводит к некорректности отклика от каждой из секций.

Данное утверждение является эвристическим (хотя есть доказательство для специальных случаев.

5. **Какие виды масштабирования бывают?**

– **Вертикальное -** добавление ресурсов в одну и ту же логическую единицу для увеличения емкости.
– **Горизонтальное -** добавление большего количества узлов в систему.

NoSQL хранилище данных может быть намного быстрее, поскольку оно использует горизонтальное масштабирование.

Тема 4. Экосистема Hadoop.

Основные вопросы темы

Основные принципы Hadoop, компоненты, примеры использования. Проблемы разработки для крупных параллельных систем

Материалы для самоподготовки

1. **Какой файл управляет отчетностью в Hadoop?**

В Hadoop файл hadoop-metrics.properties управляет отчетами.

2. **Для использования Hadoop перечислите сетевые требования?**

Для использования Hadoop список требований к сети:

- SSH-соединение без пароля

- Secure Shell (SSH) для запуска серверных процессов

3. Укажите, какой компонент хранения данных использует Hadoop?

Компонент хранения данных, используемый Hadoop, – это HBase.

4. Объясните, что такое Sqoop в Hadoop?

Для передачи данных между управлением реляционными базами данных (RDBMS) и Hadoop HDFS используется инструмент, известный как Sqoop. С помощью Sqoop данные могут передаваться из RDMS, например MySQL или Oracle, в HDFS, а также экспортироваться из файла HDFS в RDBMS.

5. Перечислите три файла конфигурации Hadoop?

Три файла конфигурации

- ядро-site.xml
- mapred-site.xml
- HDFS-site.xml
- **60) Объясните, как Hadoop Classpath играет жизненно важную роль в остановке или запуске демонов Hadoop?**
- Classpath будет состоять из списка каталогов, содержащих файлы JAR для остановки или запуска демонов.

6. Укажите, в чем разница между СУБД и Hadoop?

RDBMS	**Hadoop**
СУБД – это система управления реляционными базами данных	Hadoop – это плоская структура на основе узлов
Используется для обработки OLTP, тогда как Hadoop	В настоящее время используется для аналитической и для обработки больших данных
В СУБД кластер базы данных использует те же файлы данных, которые хранятся в общем хранилище	В Hadoop данные хранения могут храниться независимо в каждом узле обработки.
Вам необходимо предварительно обработать данные перед их сохранением	вам не нужно предварительно обрабатывать данные перед их сохранением

7. Объясните, что такое узлы хранения и вычисления?

- Узел хранения – это компьютер или компьютер, на котором находится ваша файловая система для хранения данных обработки.

- Вычислительный узел – это компьютер или машина, на которой будет выполняться ваша фактическая бизнес-логика.

8. Объясните, что такое файл последовательности в Hadoop?

Для хранения двоичных пар ключ / значение используется файл последовательности. В отличие от обычного сжатого файла, файл последовательности поддерживает разбиение, даже если данные внутри файла сжаты.

9. Объясните, как вы можете проверить, работает ли Namenode, используя команду jps?

Помимо использования команды jps, чтобы проверить, работает ли Namenode, вы также можете использовать
/etc/init.d/hadoop-0.20-namenode status.

10. Объясните, что такое трекер задач в Hadoop?

Трекер задач в Hadoop – это демон подчиненного узла в кластере, который принимает задачи из JobTracker. Каждые несколько минут он также отправляет сообщения пульса в JobTracker, чтобы подтвердить, что JobTracker все еще жив.

11. Укажите, какие демоны работают на главном и подчиненных узлах?

- Демоны запускаются на главном узле “NameNode”
- Демоны, запущенные на каждом подчиненном узле: «Task Tracker» и «Data»

12. Объясните, как можно отлаживать код Hadoop?

Популярные методы отладки кода Hadoop:

- Используя веб-интерфейс, предоставляемый платформой Hadoop
- С помощью счетчиков

Тема 5. Распределённые файловые системы.

Основные вопросы темы

Структура РФС, требования, примеры: HDFS, Google, LustreFS. HDFS: архитектура, основные узды, ограничения, основные команды.

Материалы для самоподготовки

1. Принцип действия распределенных файловых систем ?

Абстрактно описать принцип действия любой распределенной файловой системы нелегко, поэтому дальнейшие пояснения будут основываться на примере конкретной распределенной файловой системы под названием Hadoop Distributed File System (HDFS). Структура HDFS в достаточной степени представляет принцип действия распределенных файловых систем, чтобы продемонстрировать возможности их применения на уровне пакетной обработки.

HDFS и Hadoop MapReduce являются двумя ответвлениями проекта Hadoop, реализующего библиотеку Java для распределенного хранения и обработки больших объемов данных. Система Hadoop развертывается на группе серверов, обычно называемой кластером, а HDFS служит в качестве распределенной и масштабируемой файловой системы, управляющей хранением данных в кластере.

Hadoop — довольно крупный и сложный проект, поэтому мы опишем его лишь в самых общих чертах. У кластера HDFS имеются два типа узлов: единственный узел имен и несколько узлов данных. Когда файл выгружается в систему HDFS, он разбивается сначала на блоки фиксированного размера - как правило, от 64 до 256 Мбайт. Затем происходит репликация каждого блока по нескольким (как правило, трём) узлам данных, выбираемым произвольно. В узле имен отслеживается разбиение файла на блоки и местоположение каждого узла.

Такое распределение файла по многим узлам упрощает его параллельную обработку. Когда программе требуется доступ к файлу, хранящемуся в системе HDFS, она обращается к узлу имен, чтобы определить те узлы данных, где размещается содержимое этого файла. Кроме того, каждый блок реплицируется по нескольким узлам, и поэтому данные остаются доступными даже в том случае, если отдельные узлы работают в автономном режиме.

Разумеется, такой отказоустойчивости присущи свои ограничения. Так, если коэффициент репликации равен трем и одновременно выходят из строя три узла, где хранятся миллионы байтов данных, то некоторые блоки, хранящиеся в этих трех узлах, окажутся недоступными. Реализовать распределенную файловую систему совсем не просто, но, по крайней мере, вам теперь должно быть ясно, что в ней важнее всего для пользователей.

Таким образом, о распределенной файловой системе нужно знать следующее.

• Файлы распределяются по многим машинам для целей масштабируемости и параллельной обработки данных.

• Блоки файлов реплицируются по нескольким узлам для достижения отказоустойчивости.

А теперь исследуем возможности сохранения главного массива данных с помощью распределенной файловой системы.

2. Объясните, что такое NameNode в Hadoop?

NameNode в Hadoop – это узел, где Hadoop хранит всю информацию о расположении файлов в HDFS (распределенная файловая система Hadoop). Другими словами, NameNode является центральным элементом файловой системы HDFS. Он хранит записи всех

файлов в файловой системе и отслеживает данные файла через кластер или несколько компьютеров

3. **Объясните, что такое JobTracker в Hadoop? Какими действиями руководствуется Hadoop?**

В Hadoop для отправки и отслеживания заданий MapReduce используется JobTracker. Отслеживание заданий выполняется по собственному процессу JVM

Job Tracker выполняет следующие действия в Hadoop

- Клиентское приложение отправляет вакансии на трекер
- JobTracker общается в режиме имени, чтобы определить местоположение данных
- Рядом с данными или с доступными слотами JobTracker находит узлы TaskTracker
- На выбранных узлах TaskTracker, он отправляет работу
- Когда задача не выполняется, система отслеживания заданий уведомляет и решает, что делать дальше.
- Узлы TaskTracker контролируются JobTracker

4. **Объясните, что такое сердцебиение в HDFS?**

Сердцебиение относится к сигналу, используемому между узлом данных и узлом имени, а также между средством отслеживания задач и средством отслеживания заданий. Если узел имени или средство отслеживания заданий не отвечает на сигнал, то считается, что существуют некоторые проблемы с узлом данных или заданием трекер

5. **Что происходит при сбое узла данных?**

Когда происходит сбой узла данных

- Jobtracker и namenode обнаруживают сбой
- На отказавшем узле все задачи перепланированы
- Namenode реплицирует данные пользователя на другой узел

6. **Объясните, что такое спекулятивное исполнение?**

В Hadoop во время спекулятивного выполнения запускается определенное количество повторяющихся задач. На другом подчиненном узле можно выполнить несколько копий одной и той же карты или задачи сокращения с помощью спекулятивного выполнения. Проще говоря, если конкретному диску требуется много времени для выполнения задачи, Hadoop создаст дублирующую задачу на другом диске. Диск, который первым завершает задачу, сохраняется, а диски, которые не заканчивают сначала, уничтожаются.

7. **Объясните, в чем разница между входным разделением и блоком HDFS?**

Логическое разделение данных называется разделением, а физическое разделение данных – блоком HDFS.

8. **Объясните, как JobTracker планирует задачу?**

Трекер задач отправляет сообщения пульса в Jobtracker обычно каждые несколько минут, чтобы убедиться, что JobTracker активен и функционирует. Сообщение также информирует JobTracker о количестве доступных слотов, поэтому JobTracker может быть в курсе того, как можно делегировать работу кластера.

9. Объясните, что такое «карта» и что такое «редуктор» в Hadoop?

В Hadoop карта является этапом решения запросов HDFS. Карта считывает данные из местоположения ввода и выводит пару ключ-значение в соответствии с типом ввода.
В Hadoop редуктор собирает выходные данные, сгенерированные преобразователем, обрабатывает их и создает собственный конечный результат.

10. Укажите, что такое стойка для осознания?

Осведомленность о стойке – это способ, которым namenode определяет, как размещать блоки, основываясь на определениях стойки.

11. Укажите, как лучше копировать файлы между кластерами HDFS?

Лучший способ скопировать файлы между кластерами HDFS – использовать несколько узлов и команду distcp, чтобы рабочая нагрузка была общей.

12. Укажите, в чем разница между HDFS и NAS?

Блоки данных HDFS распределяются по локальным дискам всех машин в кластере, а данные NAS хранятся на выделенном оборудовании.

13. Когда Namenode не работает, что происходит с трекером работы?

Namenode – это единственная точка отказа в HDFS, поэтому, когда Namenode не работает, ваш кластер будет отключен.

14. Объясните, как осуществляется индексация в HDFS?

Hadoop имеет уникальный способ индексации. После того, как данные сохранены в соответствии с размером блока, HDFS продолжит хранить последнюю часть данных, в которой будет указано, где будет находиться следующая часть данных.

Тема 6. MapReduce: методология и технология распределённых вычислений.

Основные вопросы темы

Основные идеи MapReduce, Этапы Map – предварительной обработки и Reduce – свертки результатов. Примеры функций. Различные решения: Apache Hadoop, Erlang, в MongoDB, в CouchDB.

Материалы для самоподготовки

1. Что такое Hadoop Map Reduce?

Для обработки больших наборов данных параллельно в кластере Hadoop используется инфраструктура Hadoop MapReduce. Анализ данных использует двухэтапную карту и сокращает процесс.

2. **Как работает Hadoop MapReduce?**

В MapReduce во время фазы карты он считает слова в каждом документе, а в фазе сокращения он объединяет данные в соответствии с документом, охватывающим всю коллекцию. На этапе сопоставления входные данные делятся на разбиения для анализа по задачам сопоставления, выполняемым параллельно в среде Hadoop.

3. **Объясните, что тасует в MapReduce?**

Процесс, посредством которого система выполняет сортировку и передает выходные данные карты в редуктор в качестве входных данных, известен как случайное перемешивание.

4. **Объясните, что такое распределенный кеш в MapReduce Framework?**

Распределенный кэш – это важная функция, предоставляемая платформой MapReduce. Если вы хотите поделиться некоторыми файлами между всеми узлами в Hadoop Cluster, используется распределенный кэш. Эти файлы могут быть исполняемыми файлами jar или файлом простых свойств.

5. **Объясните, что такое объединители и когда вам следует использовать объединитель в задании MapReduce?**

Для повышения эффективности программы MapReduce используются комбинаторы. Объем данных может быть уменьшен с помощью сумматора, который необходимо передать в редукторы. Если выполняемая операция является коммутативной и ассоциативной, вы можете использовать свой код редуктора в качестве объединителя. Выполнение сумматора не гарантируется в Hadoop

6. **Объясните, каковы основные параметры картографа?**

Основными параметрами Mapper являются

- LongWritable и текст
- Текст и IntWritable

7. **Объясните, какова функция разделителя MapReduce?**

Функция разделителя MapReduce заключается в том, чтобы все значения одного ключа передавались одному и тому же редуктору, что в конечном итоге помогает равномерно распределить вывод карты по редукторам.

8. **Укажите, какие основные параметры конфигурации необходимо указать пользователю для запуска задания MapReduce?**

Пользователь каркаса MapReduce должен указать

- Расположение ввода задания в распределенной файловой системе
- Расположение вывода задания в распределенной файловой системе
- Формат ввода
- Выходной формат
- Класс, содержащий функцию карты
- Класс, содержащий функцию приведения
- JAR-файл, содержащий классы мапперов, редукторов и драйверов

9. Укажите, что такое распределенный кеш в Hadoop?

Распределенный кеш в Hadoop – это средство, предоставляемое платформой MapReduce. Во время выполнения задания оно используется для кэширования файла. Framework копирует необходимые файлы на подчиненный узел перед выполнением любой задачи на этом узле.

10. Укажите, что такое контракт API Hadoop MapReduce для класса ключа и значения?

Для класса ключа и значения существует два контракта API Hadoop MapReduce

- Значение должно определять интерфейс org.apache.hadoop.io.Writable
- Ключ должен определять интерфейс org.apache.hadoop.io.WritableComparable

11. Объясните, что происходит в текстовом формате?

В формате ввода текста каждая строка в текстовом файле является записью. Значение – это содержимое строки, а Key – байтовое смещение строки. Например, Key: longWritable, Значение: текст

12. Объясните, что такое WebDAV в Hadoop?

Для поддержки редактирования и обновления файлов WebDAV представляет собой набор расширений для HTTP. В большинстве операционных систем общие папки WebDAV могут быть смонтированы как файловые системы, поэтому можно получить доступ к HDFS как к стандартной файловой системе, открыв HDFS через WebDAV.

13. Объясните, что такое Sequencefileinputformat?

Sequencefileinputformat используется для последовательного чтения файлов. Это специальный сжатый двоичный формат файла, который оптимизирован для передачи данных между выходными данными одного задания MapReduce на вход другого задания MapReduce.

14. Объясните, что делает класс conf.setMapper?

Conf.setMapperclass устанавливает класс mapper и все вещи, связанные с заданием карты, такие как чтение данных и генерация пары ключ-значение из mapper

15. Объясните, что такое Hadoop?

Это программная платформа с открытым исходным кодом для хранения данных и запуска приложений на кластерах стандартного оборудования. Это обеспечивает огромную вычислительную мощность и большое хранилище для любого типа данных.

16. Упомянуть основные компоненты Hadoop?

Основные компоненты Hadoop включают в себя:

- HDFS
- Уменьшение карты

17. Укажите, какие компоненты данных использует Hadoop?

Компоненты данных, используемые Hadoop:

- свинья
- улей

18. Укажите, какие наиболее распространенные форматы ввода определены в Hadoop?

Наиболее распространенные форматы ввода, определенные в Hadoop:

- TextInputFormat
- KeyValueInputFormat
- SequenceFileInputFormat

19. В Hadoop, что такое InputSplit?

Он разбивает входные файлы на куски и назначает каждое разбиение мапперу для обработки.

20. Как для работы с Hadoop вы будете писать собственный разделитель?

Вы пишете пользовательский разделитель для задания Hadoop, следуя по следующему пути

- Создайте новый класс, который расширяет класс Partitioner.
- Переопределить метод getPartition
- В оболочке, которая запускает MapReduce
- Добавьте пользовательский разделитель в задание, используя набор методов Partitioner Class или – добавьте пользовательский разделитель в задание в виде файла конфигурации

21. Можно ли изменить количество создаваемых карт для работы в Hadoop?

Нет, невозможно изменить количество создаваемых картографов. Количество картографов определяется количеством входных разбиений.

22. Объясните, можно ли искать файлы с использованием подстановочных знаков?

Да, можно искать файлы с использованием подстановочных знаков.

23. Укажите, что такое использование объекта контекста?

Объект Context позволяет мапперу взаимодействовать с остальной частью Hadoop система. Он включает в себя данные конфигурации для задания, а также интерфейсы, которые позволяют ему выдавать выходные данные.

24. Укажите, каким будет следующий шаг после Mapper или MapTask?

Следующий шаг после Mapper или MapTask состоит в том, что выходные данные Mapper отсортированы, и для вывода будут созданы разделы.

25. Укажите, какое количество разделителей по умолчанию в Hadoop?

В Hadoop по умолчанию используется разделитель «Hash».

26. Объясните, какова цель RecordReader в Hadoop?

В Hadoop RecordReader загружает данные из своего источника и преобразует их в пары (ключ, значение), пригодные для чтения Mapper.

27. Объясните, как данные разделяются перед их отправкой в редуктор, если в Hadoop не определен пользовательский разделитель?

Если в Hadoop не определен пользовательский разделитель, то используемый по умолчанию разделитель вычисляет хеш-значение для ключа и назначает раздел на основе результата.

28. Объясните, что происходит, когда Hadoop порождает 50 заданий на задание и одно из заданий не выполнено?

Он перезапустит задачу еще раз на каком-либо другом TaskTracker, если задача завершится с ошибкой, превышающей установленный предел.

29. Укажите, чем Hadoop отличается от других инструментов обработки данных?

В Hadoop вы можете увеличивать или уменьшать количество картографов, не беспокоясь об объеме обрабатываемых данных.

30. Укажите, какую работу выполняет класс conf?

Класс conf задания разделяет разные задания, работающие в одном кластере. Он выполняет настройки уровня работы, такие как объявление работы в реальной среде.

31. Укажите, в каких трех режимах может работать Hadoop?

Три режима работы Hadoop:

- Псевдораспределенный режим
- Автономный (локальный) режим
- Полностью распределенный режим

32. Укажите, что делает формат ввода текста?

Формат ввода текста создаст объект строки, представляющий собой шестнадцатеричное число. Значение рассматривается как текст всей строки, а ключ рассматривается как объект строки. Картограф получит значение в виде параметра ‘text’, а ключ в качестве параметра ‘longwriteable’.

58) Укажите, сколько InputSplits сделано Hadoop Framework?

Hadoop сделает 5 сплитов

- 1 сплит для файлов 64K
- 2 сплит для 65 МБ файлов
- 2 сплит для 127 МБ файлов

Тема 7. Обработка данных в реальном времени.

Основные вопросы темы

Storm, Spark, Impal: основные идеи, сравнение достоинств и недостатков. Примеры использования.

Материалы для самоподготовки

1. **Общие свойство фреймворков потоковых вычичлений?**

– **Прикладное назначение** – все перечисленные фреймворки предназначены специально для обработки потоков Big Data «на лету», фактически в режиме реального времени.

– **Распространенность** – каждая из отмеченных сред широко используется на практике среди крупных ИТ-проектов уровня Facebook, Google, LinkedIn, eBay, Amazon и пр., в различных бизнес-кейсах и Big Data системах государственных и частных компаний.

– **Кластерная архитектура** – Apache Kafka Streams, Spark Streaming, Flink, Storm и Samza являются средствами разработки распределенных приложений и потоковой обработки данных, хранящихся на разных физических и виртуальных узлах одного или нескольких кластеров.

– **Распараллеливание задач** – каждый инструмент поддерживает параллелизм вычислений, распределяя их по направленной графовой модели потоковых обработчиков, называемой DAG-топологией (Directed Acyclic Graph).

– **Отказоустойчивость** – надежность распределенных вычислений обеспечивается специальными механизмами восстановления в случае сбоя (контрольные точки или транзакции), которые позволяют вернуться к прерванной задаче по обработке данных и запустить ее заново на этом же самом или новом узле кластера.

2. Apache Storm что это?

Apache Storm (Сторм, Шторм) часто употребляется в контексте других BigData инструментов для распределенных потоковых вычислений в реальном времени (Real Time, RT): Spark Streaming, Kafka Streams, Flink и Samza. Однако, если Apache Spark и Flink по функциональным возможностям и составу компонентов еще могут конкурировать между собой, то сравнивать с ними Шторм, предназначенный только для относительно простой по логике распределенной обработки потоковых событий, не совсем объективно. Более целесообразно выбирать между Сторм и Apache Samza, что мы рассмотрим в нашей следующей статье. А сегодня поговорим о том, где используется Apache Storm и в каких случаях следует применять именно этот Big Data фреймворк потоковых RT-вычислений.

3. Когда нужен Storm?

Storm будет отличным выбором для реализации высокоскоростной системы обработки событий, обеспечивающей инкрементные вычисления, в т.ч. по требованию. Однако, если в рамках Шторм необходимо обеспечить сохранение состояния (stateful) и в точности однократную доставку сообщений (exactly once), следует использовать Trident API, который позволяет работать с микропакетами, как Apache Spark.

4. Что Такое Spark?

Apache Spark — это фреймворк с открытым исходным кодом, который в основном используется для анализа Big Data, машинного обучения и обработки данных в реальном времени. Фреймворк в общем и целом обеспечивает полнофункциональный интерфейс для программистов и разработчиков – этот интерфейс отлично помогает в решении различных задач кластерного программирования и машинного обучения.

5. Каковы некоторые из самых заметных особенностей apache spark?

Три особенности – **скорость, мультиформатная поддержка** и **встроенные библиотеки**.

Поскольку существует минимальное количество сетей, обрабатывающих данные, движок Spark может достигать потрясающих скоростей, особенно по сравнению с Hadoop. Кстати, скорость является очень важным аспектом, если темой является именно Spark Streaming.

В дополнение к этому, Spark поддерживает множество источников данных (так как он использует Spark SQL для их интеграции) и имеет огромный выбор различных, стандартных библиотек, которые могут использовать разработчики Big Data.

6. Что такое SCC?

SCC означает “*Spark Cassandra Connector*”. Это инструмент, которые использует Spark для доступа к информации (данным), расположенным в различных базах данных Cassandra.

7. Что такое RDD?

RDD означает “*Resilient Distribution Datasets*”. Это операционные элементы, которые при запуске могут работать параллельно друг другу. Всего существует два известных типа RDD – распараллеленные коллекции и наборы данных Hadoop. RDD также поддерживает два типа операций – **действия** и **трансформации.**

8. Что такое неизменность (Immutability)?

Как становится понятно из названия, если предмет **неизменен,** то он не может быть изменён после того, как он полностью создан и уже имеет значение.

Spark (в качестве фреймворка) имеет такую функцию. Однако она не применима к процессам сбора данных – только для их установленных значений.

9. Что Такое YARN?

YARN является одной из ключевых функций Spark. Она в основном нацелена на управление ресурсами, но также может использоваться для работы с кластерами Spark – благодаря своей невероятной масштабируемости.

Тема 8. Массово-параллельная структура - Massive Parallel Processing.

Основные вопросы темы

Масштабирование реляционных баз данных. Параллельное выполнение запросов к БД. Архитектура Hub and Spoke.

Материалы для самоподготовки

10. Масштабируемость на уровне пакетной обработки ?

Словом масштабируемость слишком часто разбрасываются, поэтому выясним тщательно, что же оно обозначает в контексте информационных систем. Масштабируемость — это способность системы поддерживать свою производительность при увеличении нагрузки вводом дополнительных ресурсов. Нагрузка в контексте больших данных - 4 - обозначает сочетание общего объема имеющихся данных с объемом ежедневно получаемых новых данных, количеством запросов, обслуживаемых приложением каждую секунду, и т.д. и т.п. Еще важнее, что масштабируемая система является линейно масштабируемой. В частности, линейно масштабируемая система способна поддерживать свою производительность при увеличении нагрузки за счет ввода дополнительных ресурсов пропорционально растущей нагрузке. И хотя нелинейно

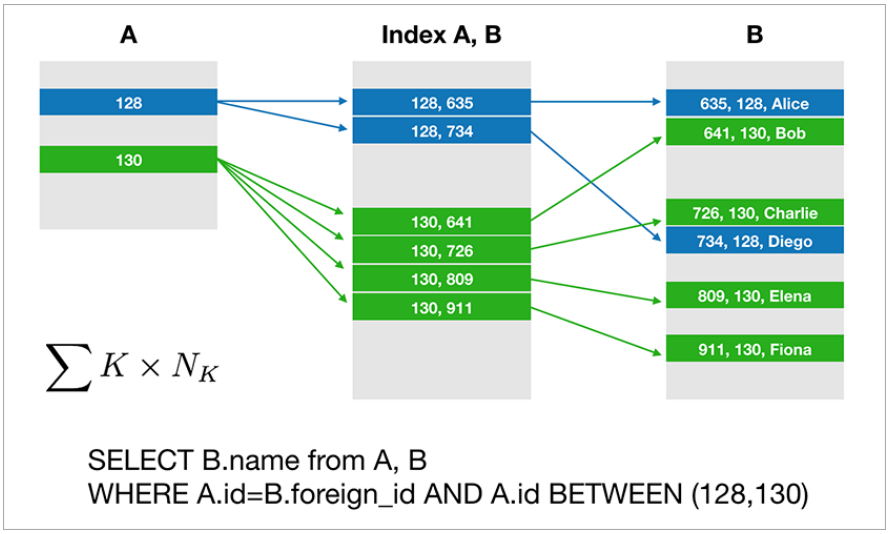
масштабируемая система также считается масштабируемой, пользы от нее не очень много. Допустим, количество требующихся машин находится в квадратической зависимости от нагрузки на систему. Затраты на эксплуатацию такой системы могут со временем значительно возрасти. Так, если нагрузка на систему увеличится в 10 раз, то затраты на ее эксплуатацию — в 100 раз.

11. Что масштабируемость не означает ?

Как ни странно, масштабируемая система совсем не обязательно должна быть в состоянии повышать свою производительность вводом дополнительных машин. Допустим, имеется вебсайт, обслуживающий статическую HTML-страницу. Допустим также, что каждый веб-сервер способен обслужить 1000 запросов в секунду с допустимой задержкой 100 миллисекунд. Сократить эту задержку обслуживания веб-страницы не удастся вводом дополнительных машин, поскольку отдельный запрос не распараллеливается и должен быть обслужен одной машиной. Но при увеличении количества запросов в секунду можно масштабировать веб-сайт, введя дополнительные веб-серверы, чтобы распределить нагрузку на обслуживание HTML-страницы. С помощью алгоритмов, допускающих распараллеливание, можно практически повысить производительность, введя дополнительные машины, но усовершенствования этих алгоритмов способны свести на нет эффект от ввода дополнительных машин. Дело в том, что ввод дополнительных машин влечет за собой увеличение расходов на эксплуатацию и связь. Таким образом, мы подготовили почву для обсуждения MapReduce — парадигмы распределенных вычислений, с помощью которой можно реализовать уровень пакетной обработки. При дальнейшем рассмотрении этой парадигмы следует иметь в виду ее линейную масштабируемость. Так, если размер главного массива данных увеличивается вдвое, то в той же степени возрастает и количество серверов, на которых можно строить пакетные представления при той же самой задержке обработки запросов.

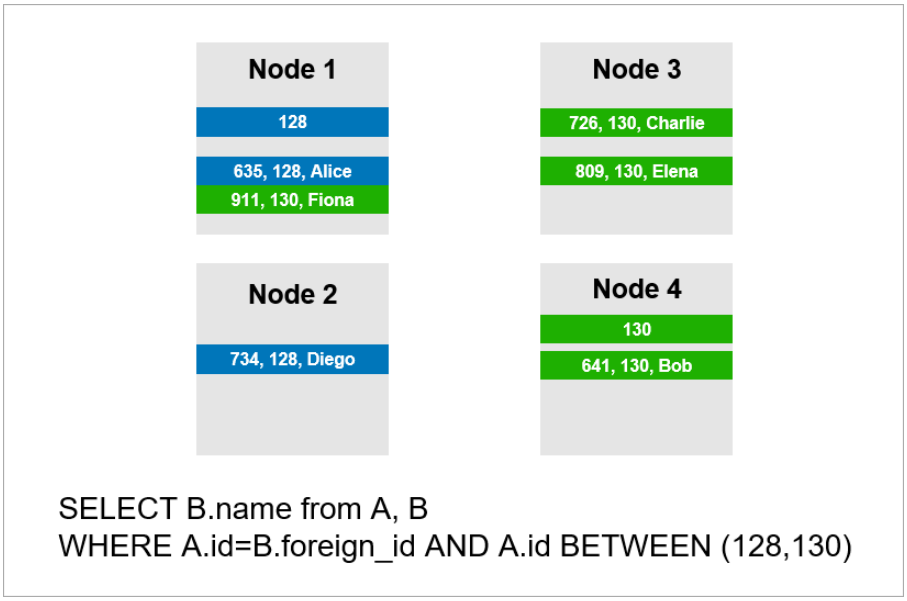
12. Как масштабируется реляционная БД?

Большинство баз данных, с которыми разработчики привыкли работать, поддерживают реляционную алгебру. Данные хранятся в таблицах и временами нужно соединить данные из разных таблиц путем операции JOIN. Рассмотрим пример БД и простого запроса к ней.



Предполагаем, что A.id — это первичный ключ с кластерным индексом. Тогда оптимизатор построит план, который скорее всего вначале выберет нужные записи из таблицы A и затем возьмет из подходящего индекса (A,B) соответствующие ссылки на записи в таблице B. Время выполнения этого запроса растет логарифмически от количества записей в таблицах.

Теперь представьте, что данные распределены по четырем серверам кластера и вам нужно выполнить тот же самый запрос:



Если СУБД не хочет просматривать все записи всего кластера, то она вероятно попробует найти записи с A.id равным 128, 129, или 130 и найти для них подходящие записи из таблицы B. Но если A.id не является ключом шардирования, то СУБД заранее не может знать, на каком сервере лежат данные таблицы A. Придется все равно обратиться ко всем серверам, чтобы узнать, есть ли там подходящие под наше условие записи A.id. Потом каждый сервер может сделать JOIN внутри себя, но этого не достаточно. Видите, запись на ноде 2 нам нужна в выборке, но там нет записи с A.id=128?

Если ноды 1 и 2 будут делать JOIN независимо, то результат запроса будет неполным — часть данных мы не получим.

Поэтому для выполнения этого запроса каждый сервер должен обратиться ко всем остальным. Время выполнения растет квадратично от количества серверов. (Вам повезет, если вы сможете шардировать все таблицы по одному и тому же ключу, тогда все сервера обходить не нужно. Однако, на практике это малореально — всегда будут запросы, где требуется выборка не по ключу шардирования.)

Таким образом, операции JOIN масштабируются принципиально плохо и это фундаментальная проблема реляционного подхода.

13. Что означает параллельный запрос?

Параллельный запрос - это метод, используемый для увеличения скорости выполнения запросов SQL путем создания нескольких процессов запросов, которые делят рабочую нагрузку оператора SQL, и выполняют его параллельно или одновременно.

Поскольку каждый процесс одновременно работает над чем-то другим, это значительно сокращает общее время выполнения оператора SQL. Это очень полезно для систем с несколькими процессорами, которые могут работать в процессах.

14. Какие ограничения есть на параллельное выполнение запросов (на примере PostgreSQL)?

- Не включайте параллельное выполнение, если все ядра уже заняты, иначе другие запросы будут тормозить.
- Самое главное, параллельная обработка с высокими значениями WORK_MEM задействует много памяти — каждое хэш-соединение или сортировка занимают память в объеме work_mem.
- Запросы OLTP с низкой задержкой невозможно ускорить параллельным выполнением. А если запрос возвращает одну строку, параллельная обработка его только замедлит.
- Разработчики любят использовать бенчмарк TPC-H. Может, у вас есть похожие запросы для идеального параллельного выполнения.
- Только запросы SELECT без предикатной блокировки выполняются параллельно.
- Иногда правильная индексация лучше последовательного сканирования таблицы в параллельном режиме.
- Приостановка запросов и курсоры не поддерживаются.

- Оконные функции и агрегатные функции упорядоченных наборов не параллельны.
- Вы ничего не выигрываете в рабочей нагрузке ввода-вывода.
- Параллельных алгоритмов сортировки не бывает. Но запросы с сортировками могут выполняться параллельно в некоторых аспектах.
- Замените CTE (WITH …) на вложенный SELECT, чтобы включить параллельную обработку.
- Обертки сторонних данных пока не поддерживают параллельную обработку (а могли бы!)
- FULL OUTER JOIN не поддерживается.
- max_rows отключает параллельную обработку.
- Если в запросе есть функция, не помеченная как PARALLEL SAFE, он будет однопоточным.
- Уровень изоляции транзакции SERIALIZABLE отключает параллельную обработку.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Основная литература

1. Макшанов, А. В. Большие данные. Big Data / А. В. Макшанов, А. Е. Журавлев, Л. Н. Тындыкарь. — 2-е изд., стер. — Санкт-Петербург : Лань, 2022. — 188 с. — ISBN 978-5-8114-9690-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/198599. — Режим доступа: для авториз. пользователей.

2. Просто BIG DATA . — Санкт-Петербург : Страта, 2019. — 148 с. — ISBN 978-5-907127-29-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/141885. — Режим доступа: для авториз. пользователей.

3. Радченко, И. А. Технологии и инфраструктура Big Data : учебное пособие / И. А. Радченко, И. Н. Николаев. — Санкт-Петербург : НИУ ИТМО, 2018. — 52 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/136430— Режим доступа: для авториз. пользователей.

Дополнительная литература:

1. Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP : учеб. пособие по спец. 071900 "Информ. системы и технологии" направления 654700 "Информ. системы" / А. А. Барсегян [и др.]. - 2-е изд., перераб. и доп. - Санкт-Петербург : БХВ-Петербург, 2008.

2.Федин, Ф. О. Анализ данных. Часть 1. Подготовка данных к анализу [Электронный ресурс] : учебное пособие / Ф. О. Федин, Ф. Ф. Федин. — Электрон. текстовые данные. — М. : Московский городской педагогический университет, 2012. — 204 с. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/26444.html

3.Федин, Ф. О. Анализ данных. Часть 2. Инструменты Data Mining [Электронный ресурс] : учебное пособие / Ф. О. Федин, Ф. Ф. Федин. — Электрон. текстовые данные. — М. : Московский городской педагогический университет, 2012. — 308 с. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/26445.html

4. Системы поддержки принятия решений : учебник и практикум для бакалавриата и магистратуры / В. Г. Халин [и др.] ; под редакцией В. Г. Халина, Г. В. Черновой. — Москва : Издательство Юрайт, 2019. — 494 с. — (Высшее образование). — ISBN 978-5-

534-01419-8. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://www.biblio-online.ru/bcode/432974

5. Липатова Светлана Валерьевна. Системы принятия решений : учеб.-метод. пособие / Липатова Светлана Валерьевна; УлГУ, ФМИиАТ. - Ульяновск : УлГУ, 2016. – URL: ftp://10.2.96.134/Text/Lipatova2016.pdf