Тема 3. Обработка больших данных с помощью Hadoop

Лекция 3.1. Экосистема Apache Hadoop

Apache Hadoop — фреймворк с открытым исходным кодом для распределённого хранения и обработки больших данных. Создан Дагом Каттингом (первоначально как часть проекта Nutch) на основе двух публикаций Google: GFS (2003) и MapReduce (2004). В 2006 году выделен в отдельный проект Apache, в 2008 году выиграл конкурс по сортировке — отсортировал 1 терабайт за 209 секунд на кластере из 910 узлов.

Четыре ядра Hadoop

— Hadoop Common: общие утилиты, библиотеки и абстракции файловых систем, используемые остальными модулями.

— HDFS: распределённая файловая система (подробно рассмотрена в теме 2).

— YARN: менеджер ресурсов кластера, появившийся в Hadoop 2.0 (2013).

— MapReduce: фреймворк параллельной обработки данных.

Компоненты расширенной экосистемы

— Apache Hive: SQL-интерфейс для данных в HDFS. HiveQL-запросы транслируются в MapReduce (или Tez/Spark). Facebook обрабатывает с Hive около 700 ТБ данных ежедневно.

— Apache Pig: язык потоковой обработки Pig Latin для ETL. Скрипты транслируются в MapReduce. Подходит для нерегулярных преобразований данных.

— Apache ZooKeeper: сервис координации для распределённых приложений — выборы лидера, распределённые блокировки, хранение конфигурации. Используется Hadoop, HBase, Kafka.

— Apache Sqoop: перенос данных между HDFS/Hive и реляционными СУБД (MySQL, Oracle, PostgreSQL). Поддерживает параллельный импорт/экспорт.

— Apache Flume: надёжный сбор, агрегация и перемещение потоков журналов и событий в HDFS.

— Apache Oozie: планировщик рабочих процессов для Hadoop. Цепочки задач описываются в формате DAG (directed acyclic graph) на XML.

Лекция 3.2. Модель программирования MapReduce

MapReduce — парадигма программирования, разделяющая обработку на две функции: Map (параллельное преобразование) и Reduce (агрегация результатов). Ключевое свойство: обе функции не имеют состояния и не зависят друг от друга внутри одной фазы, что обеспечивает безопасный параллелизм.

Фазы выполнения задания

— Input Splits: входной файл в HDFS делится на сплиты, обычно совпадающие с блоками. Каждый сплит обрабатывает один mapper — это реализует принцип локальности данных.

— Map: функция применяется к каждой входной записи независимо. Принимает пару (ключ, значение), выдаёт ноль или более промежуточных пар (ключ', значение'). Запускается параллельно на всех узлах, где есть данные.

— Combine (опционально): локальная мини-редукция на узле mapper до передачи данных по сети. Снижает объём сетевого трафика. Применима только к коммутативным и ассоциативным операциям.

— Shuffle and Sort: автоматическая фаза. Промежуточные пары сортируются по ключу и группируются: все значения с одинаковым ключом направляются одному reducer. Это самая дорогая по сетевому трафику фаза.

— Reduce: функция получает ключ и итерируемый список значений. Выполняет агрегацию (суммирование, поиск максимума, подсчёт) и выдаёт финальный результат.

— Output: результат записывается в HDFS в виде пар (ключ, значение).

Практические примеры задач MapReduce

— Word Count: map выдаёт (слово, 1) для каждого слова; reduce суммирует единицы по каждому слову. Классический учебный пример.

— Инвертированный индекс для поисковика: map выдаёт (слово, документ); reduce собирает список документов для каждого слова.

— Вычисление PageRank: итеративный алгоритм ранжирования веб-страниц по входящим ссылкам. Каждый шаг — одно задание MapReduce.

— Матричное умножение больших разрежённых матриц, используемых в рекомендательных системах.

Ограничения MapReduce

Каждый шаг записывает результат на диск, что делает итерационные алгоритмы (характерные для машинного обучения) крайне медленными. Задача кластеризации с 10 итерациями требует 20 операций чтения-записи на HDFS. Отсутствие потоковой обработки исключает сценарии реального времени. Эти ограничения стали основной мотивацией создания Apache Spark.

Лекция 3.3. YARN — архитектура и функции

В Hadoop 1.x JobTracker совмещал управление ресурсами и координацию задач. При кластере более 4000 узлов он становился узким местом. YARN (Yet Another Resource Negotiator), появившийся в Hadoop 2.0, разделяет эти функции и превращает кластер в

универсальную платформу, на которой могут работать любые фреймворки обработки данных.

— ResourceManager: единственный компонент, управляющий ресурсами всего кластера. Состоит из Scheduler (распределяет CPU и RAM по приложениям) и ApplicationsManager (принимает заявки на запуск приложений и отслеживает их жизненный цикл).

— NodeManager: агент на каждом рабочем узле. Запускает контейнеры по команде ResourceManager, отслеживает потребление CPU и памяти, сообщает о нарушениях лимитов.

— ApplicationMaster: компонент уровня конкретного приложения (один AM на приложение). Запрашивает у ResourceManager контейнеры, распределяет задачи по полученным контейнерам. Это разгружает ResourceManager от специфики конкретных фреймворков.

— Container: единица ресурсов (набор CPU и RAM). Контейнер запускается на NodeManager и существует только на время выполнения задачи.

Поверх YARN работают не только MapReduce, но и Apache Spark, Apache Flink, Apache Storm и другие фреймворки. Это позволяет разным приложениям совместно использовать ресурсы одного кластера, что значительно повышает утилизацию оборудования.

Лекция 3.4. Форматы хранения данных в Hadoop

Выбор формата файла критически влияет на производительность аналитических задач. Правильный формат сокращает время выполнения запросов в десятки раз и снижает стоимость хранения.

— Text/CSV: универсальный, читаемый человеком. Нет сжатия схемы, нет разбиения на блоки (splittable только при использовании bzip2). Приемлем для небольших наборов данных и при обмене с внешними системами.

— JSON: гибкий, самоописывающий. Каждый документ содержит имена всех полей, что даёт огромную избыточность при миллиардах строк. Формат JSONL (JSON Lines) лучше поддаётся параллельному чтению.

— Apache Avro: бинарный построчный формат со схемой. Схема хранится отдельно (в Confluent Schema Registry при работе с Kafka) или в начале файла. Оптимален для потоковой передачи и ETL-конвейеров.

— Apache Parquet: бинарный столбцовый формат. Для запроса «SELECT col1, col2 FROM huge_table» читаются только два столбца, остальные не затрагиваются. Встроенное сжатие Snappy или Gzip. Стандарт де-факто для хранения в Data Lake.

— Apache ORC: столбцовый формат, разработанный специально для Hive. Поддерживает predicate pushdown — фильтры применяются ещё до чтения данных. Часто эффективнее Parquet при работе с Hive.

Лекция 3.5. Apache Hive: SQL-аналитика на больших данных

Hive позволяет аналитикам, владеющим SQL, работать с данными в HDFS без написания кода на Java. HiveQL — диалект SQL с расширениями для работы с большими данными. Исторически Hive транслировал запросы в задачи MapReduce; современные версии используют Apache Tez или Spark как более быстрые execution engine.

Hive Metastore

Metastore — центральный каталог метаданных. Хранит схемы таблиц (имена и типы столбцов), физическое расположение данных в HDFS, формат файлов, параметры сериализации/десериализации (SerDe). Metastore используется совместно несколькими инструментами — Hive, Spark, Impala, Presto — что обеспечивает единое пространство имён таблиц.

Партиционирование и бакетирование

Партиционирование физически разделяет данные по значению одного или нескольких столбцов. Таблица событий, партиционированная по дате, хранит данные за каждый день в отдельном каталоге HDFS. Запрос за конкретный день читает только соответствующий каталог — partition pruning. Бакетирование делит данные внутри партиции на фиксированное число файлов по хэшу ключа. Ускоряет JOIN (если обе таблицы забакетированы по ключу соединения) и операции sampling.

Лекция 3.6. Развёртывание и мониторинг кластера Hadoop

— Standalone mode: Hadoop работает как единый Java-процесс без HDFS. Для разработки и отладки MapReduce-кода.

— Pseudo-distributed mode: все компоненты Hadoop запущены на одной машине как отдельные процессы, данные хранятся в HDFS. Для тестирования с реальной файловой системой.

— Fully distributed mode: кластер из множества узлов. NameNode и ResourceManager — на выделенных мастер-узлах. DataNode и NodeManager — на всех рабочих узлах.

Ключевые конфигурационные файлы

— core-site.xml: базовая конфигурация — адрес NameNode (параметр fs.defaultFS), временная директория.

— hdfs-site.xml: фактор репликации (dfs.replication), пути хранения данных NameNode и DataNode.

— mapred-site.xml: выбор execution framework (yarn), параметры памяти для mapper и reducer.

— yarn-site.xml: адрес ResourceManager, настройки памяти NodeManager, параметры планировщика (FIFO, Capacity, Fair Scheduler).

Мониторинг

— HDFS Web UI на порту 9870 (NameNode): состояние файловой системы, список DataNode, использование дискового пространства, сводка по блокам.

— YARN ResourceManager UI на порту 8088: запущенные и завершённые приложения, потребление ресурсов, очереди планировщика.

— Cloudera Manager и Apache Ambari: коммерческие и открытые платформы управления кластером с дашбордами, предупреждениями и автоматической настройкой.