第十二周总结

学习笔记

12.1 大数据原理概述

1. 大数据狭义指的是hadoop，hive这一套，广义则包括机器学习，数据应用，数据挖掘等等。

2. Google 2004年的论文奠定了大数据基础，分别是分布式文件系统GFS，大数据分布式计算框架Map Reduce，和NoSql数据库系统BigTable

3. 随着技术的发展，YARN从Hadoop中分离出来成为了一个独立的资源调度框架

4. 除了大数据的离线流程之外，后续又出现了flink，storm，spark stream等框架用于实时的大数据计算。

5. 大数据起初用于搜索公司优化搜索引擎，后来几乎所有的互联网公司都会收集大量数据，并借助大数据系统进行数据挖掘，数据分析和运营。

6. 机器学习也是数据挖掘的一种，随着算力的提升应用越来越广泛

7. 大数据应用场景有辅助诊疗，AI外语老师，智能解题，舆情监控分析，大数据风控，新零售无人收银，无人驾驶

12.2 分布式文件系统HDFS

1. HDFS原理和RAID有一定的类似

2. Namenode的小细节：就是记录了一个个文件包含了哪些块，有几个备份，分别在哪几个服务器上

3. HDFS设计目标：

    1. 以流式数据访问

    2. 存储超大文件

    3. 运用于商用硬件集群

    4. 节点失效是常态，任何一个节点失效，都不影响HDFS的正常使用

4. 不适合HDFS的场景：

    1. 低延迟的数据访问

    2. 大量小文件：超过namenode处理能力

    3. 多用户随机写入修改文件

5. HDFS写过程：访问NameNode获取dataNode节点和分片，备份等信息 -> 访问dataNode并写入 -> dataNode在写入期间同时往其他节点拷贝 -> 写入文件成功后再发消息告诉nameNode写入成功

6. HDFS读过程：访问nameNode获取目标文件地址 -> 随机选择或者由框架选择目标dataNode -> 读取数据

7. NameNode持久化：操作日志+FsImage

8. 目前nameNode是主主复制的架构，用zk进行协调，高可用

9. HDFS可以跨机架进行复制，进一步保障高可用

12.3 大数据计算框架MapReduce - 编程模型

1. MapReduce：大规模数据处理

    1. 处理海量数据

    2. 实现上百上千CPU并行处理

    3. 移动计算比移动数据更划算（把计算程序分发到数据服务器上进行处理）

    4. 分而治之

2. MapReduce编程模型

    1. Map和reduce的输入输出都是Key value对，其中，map输入的key是偏移量，value是一行数据，输出是真正的k,v键值对，reduce会把map输出的k,v 键值对进行处理，其中，这个value是对map输出的键值对的合并结果，同一个key会发送给同一个reducer处理。

    2. 大数据处理的核心关键：对于mapper的输出，相同的key通过shuffle操作，交给同一个reducer去处理

12.4 大数据计算框架MapReduce - 架构

1. 基本实现框架：首先有一个服务器跑了JobTracker进程，在收到计算任务以后JobTracker会分解任务并生成执行树，树上每个节点是一个TaskInProcess，每个TaskInProcess会下发分配任务给DataNode节点并启动TaskTracker进程进行计算。JobTracker在分配任务的时候，会优先给DataNode分配本地数据对应的数据的分片计算任务

2. 适合MapReduce的计算：TopK, K-means, Bayes, SQL

3. 不适合MapReduce的计算：Fibonacci

4. MapReduce通过Key对数据进行分区，进而把数据按我们自己的需求来分发。就是通过这种方式来实现，把同一个key的数据交给同一个reducer去处理的逻辑。mapreduce里面默认的缺省分区代码就是简单的key的哈希码取余得到reducer编号。

5. 作业控制

    1. 作业抽象成三层：作业监控层（JIP），任务控制层（TIP），任务执行层。

    2. 任务可能会被尝试多次执行，每个任务实例被称作Task Attempt（TA）

    3. TA成功，TIP会标注该任务成功，所有TIP成功，JIP成功

6. JobTracker容错：一般JobTracker挂了任务就失败了，虽然可以通过日志恢复一部分进度，但是一般不这么做。

7．TaskTracker容错

1. 超时：TaskTracker十分钟不报告心跳，就会被JobTracker从集群移除

2. 黑灰名单：部署性能监控脚本或者直接加黑灰名单，使得一些慢的服务器不被分配任务，防止拖慢整个集群

8. Task容错

1. 允许部分任务失败，可以设置允许失败率，默认是0

2. Task有TIP监控，失败任务可以多次尝试，可以设置最大尝试次数

9. Record容错

1. K，V超大导致OOM：可以设置截断长度

2. 异常数据引起BUG：可以设置失败几次以后跳过失败的记录

12.5 大数据集群资源管理系统Yarn

1. YARN: yet another resource negotiator。 yarn其实是从早期的MapReduce框架中单独分离出来的资源管理器，类似于JobTracker和TaskTracker的作用，有了yarn以后，同一个datanode集群可以同时运行各种各样的大数据计算引擎

2. Yarn架构：有一个服务器运行资源管理器进程，每个datanode都会运行节点管理器进程，一主多从，和MapReduce，hdfs的管理方式都比较像

3. Yarn启动任务的过程：应用程序申请启动任务 -> Yarn分配一个空闲的datanode容器 -> 在空闲dataNode中启动applicationMaster(JobTacker移除资源调度以后的角色) -> ApplicationMaster向Yarn资源管理器申请计算节点 -> YARN分配资源节点 -> applicationMaster和被分配的资源节点通信并植入运行代码

4. 由于Yarn的出现，一个HDFS集群可以同时运行很多种大数据框架，mapreduce，spark，hive等等

12.6 大数据仓库hive

1. 大数据处理任务90%都是取数任务，为了降低MapReduce的使用门槛，人们希望可以把SQL自动转化成MapReduce任务，hive就是这个背景下的产物

2. hive架构：

1. 包含client，Metastore，compiler，driver，hadoop

2. 通过client提交任务，由driver完成核心计算。metastore就像namenode一样存储元数据，记录表结构信息，字段名，字段类型等。driver会从metastore获取元数据信息，然后通过compiler把提交的任务编译成一个MapReduce执行计划，执行计划提交给Hadoop执行，得到的就是最终结果了

3. Hive的编译器是核心：

    1. Parser：把SQL编译成抽象语法树（AST）
    2. SemanticAnalyzer：把抽象语法树转化为查询块（QB）
    3. Logical Plan Generator：把QB转化为执行计划（Logical Plan）
    4. LogicalOptimizer：重写执行计划，带入更多优化后的计划
    5. Physical Plan Generator：将逻辑执行计划转化为物理执行计划（M/R jobs）
    6. Physical Optimizer：适应性Join策略调整

4. Hive会内置各种各样的Operator，这些operator实质上就是Map或者reduce的逻辑，hive的关键就是把语法转换为一个个的operator

5. MapJoin特性：当join的其中一个表比较小的时候，可以用mapJoin的方式把数据多的表的数据分发给数据少的表，在map阶段完成join，可以省资源。

个人总结

本周主要是大数据相关的知识和应用，对我而言，这是一个很好的补充，我平时工作以服务端开发为主，偶尔会用hive写点hql，udf，在一些特殊的场景也拿hdfs做过存储，之前还研究过kuberflow的使用。不过，对于大数据底层的实现原理总感觉像是蒙着一层纱，感觉不清晰。本周课程让我理解了大数据框架的运作原理，以及部分的实现细节，增强了我对技术本身的认知和把控，收获还是颇丰的。