大数据学习路径

概要

大数据知识体系，从数据的流向来看，主要包含下面的几个方面，数据的采集、传输、计算、存储和分析，每个方向都有对应的框架组件，组成了整个大数据生态。下面的学习路径则是根据这几个方向，选出生产环境下最常用的组件。

前期建议先从视频中学习，效率较高，理解并掌握基础使用，对每个方向组件的作用及常用情景有了解，形成一个系统的大数据思维。后期在实际生产中，涉及到具体组件后，可以再购买相关的书籍深入学习。

视频的选择，建议使用同一套视频，这样搭建的集群环境，使用的数据都具有延续性，可以减少很多环境问题。

推荐使用的全套视频：

```
下载链接：https://pan.baidu.com/s/18Feqa_63640xPBOfYJ8Ttg
提取码：9bnr

视频包含了下面所有阶段的框架及项目练习，甚至还有很多其他框架，其他框架建议使用到的时候再看。
```

而书籍的话，有需要深入看的框架，已经有在每个框架后有推荐书籍。

在学习过程中，慢慢会形成几种大数据的思想：

- 分布式思想
- 数据及元数据思想
- NoSQL数据库思想
- MR计算思想
- 计算、数据高可用思想
- 数仓思想（分层建模）
- 集群间交互的主从思想
- 其他（暂时想到这些，后续可补充...）

阶段一：前置知识

JavaSE

MySQL

JDBC

Maven

Linux

- Linux的基本操作：文件查看、进程查看、权限变更、用户切换
- 集群搭建：使用虚拟机搭建集群，通信成功

Shell

- 任务流：变量、运算符、条件判断、流程控制、自定义函数
- 文本处理：常用的工具awk、cut、sed等等

Redis

- NoSQL数据库：Redis的基本操作
- Redis的wuda数据类型及基本操作Api

阶段二：Hadoop基础

Hadoop是大数据体系的基石。整个大数据的其他框架，都源于谷歌的三篇论文，而Hadoop是三篇论文的基础实现。这部分的内容需要重点理解。掌握了Hadoop和Hive后，可以胜任数仓岗。

Hadoop

hadoop起源于谷歌的三篇论文，也是大数据所有框架的理论基础。

- Google File System：衍生出了开源的大数据的存储文件系统 HDFS
- Google MapReduce：衍生出了开源的大数据的计算框架 MapReduce
- Google BigTable：衍生出开源的大数据的NoSQL数据库 HBase

参考

> https://blog.csdn.net/wyz0516071128/article/details/80629055?utm_medium=distribute.pc_relevant_download.none-task-blog-baidujs-1.nonecase&depth_1-utm_source=distribute.pc_relevant_download.none-task-blog-baidujs-1.nonecase

HDFS

这是一个分布式的文件系统，通过集群存储数据。它保证了数据的多副本、分片等等。需要学习的内容有：

- NameNode和DataNode的工作机制，通信机制。
- hdfs的api操作

MapReduce

这是大数据最经典、最基本的分布式计算处理思想，而MR程序是这个思想的最初实现方式，后面的spark、flink等，也都是在这个思想上做的计算框架。

- 手写MR程序：理解输入输出的数据结构，不同的输入文件的方式。
- 理解Map和Reduce两阶段的处理思想
- 完全理解数据在整个MR中被处理过程中的Shuffle机制

Yarn

这是一个集群运行MR程序等任务时，负责给任务分配资源的组件，即管理整个集群的资源。

- Yarn的工作机制，MapReduce程序在Yarn上的提交流程

其他（重要）

- 掌握在Linux下搭建并操作Hadoop集群，Hadoop各个配置文件的配置，脚本编写
- 存在HDFS的文件存储格式、压缩方式（这部分在Hive中会使用到）
- MapReduce的参数调优。
- Yarn的三种调度器。

ZooKeeper

由于集群是分布式，多机器。为了保证多个机器间的通信，维持机器间的关系（如主从等），公共信息一致，引入了ZooKeeper这个中间件，协调集群间的机器，为集群的分布式服务。

需要掌握的点：

- ZooKeeper的数据结构、选举机制
- ZooKeeper的监听器原理
- 客户端的命令行操作，数据的查询，API的使用。

Hive

Hadoop出来后，使得大数据在生产环境下可以大规模使用。但处理大数据需要写完整的MapReduce代码，虽然模板化，但繁杂且冗余。为了降低开发门槛、提升开发效率，Facebook研发出了Hive，开发人员只要写HiveSQL（语法大体和SQL相同），通过底层的解析器等，可以转成MapReduce代码，实现和MR程序同样的数据处理逻辑。

所以学习Hive的过程，需要理解每个HiveSQL对应MR流程中的那个步骤。

Hive是大数据开发需要掌握的最基本工具。

需要掌握的点：

基本操作

- 集群中安装部署Hive
- Hive的数据类型：基本数据类型、集合数据类型（注意区别Struct 和 Map两种类型）
- 掌握DDL操作，理解管理表和外部表，理解什么是分区表，理解元数据和数据的映射。
- 掌握DML操作，hdfs上数据如何导入到（映射到）hive表，hive表数据（映射的数据）如何导出。

HiveSQL

- 类似SQL
    - SQL一样的列查询、where语句、分组
    - Join语句：区别多种类型的Join
    - 排序：区别Order By、Sort By、Distribute By、Cluster By 四种排序
    - 分桶表：分桶的含义及语句，分桶抽样查询
- 行列转换

  相对抽象，但是在某些场景下很有用
- 窗口函数

  窗口函数是HiveSQL在处理大数据时候，表达更强于SQL的一种函数。在大数据中非常常用，某些SQL很难实现的场景，使用开窗函数后，变得很简单。所以很重要！

- 自定义函数

  包含UDF和UDTF。某些数据处理问题，Hive自带的函数无法实现的使用，我们需要使用Java，自定义函数的处理逻辑后，上传jar包。如：ip的解析为具体地址，版本号映射为分值等等。

和Hadoop结合

- HiveSQL—>MR

  前面已经说过，Hive其实就是在Hadoop的基础上，封装的一个更易用的工具。所以需要理解Hive中的HQL语句和MapReduce程序的关联。

- Hive表映射HDFS文件

  同时Hive操作的数据，存储在HDFS上，所有在建立Hive表时，需要映射到实际HDFS的数据。因此需要关注文件的存储格式、压缩格式。

- HiveSQL语句的调优
  - 纯SQL语句调优
  - 根据Hadoop中的存储及计算的原理为依据的调优

项目练习

学完上面的内容后，一定要练习一个小项目，巩固Hive的基础知识，熟悉HiveSQL语句。

但是由于还没有学习其他的框架，如Kafka、Flume、Sqoop等，所以数仓项目暂时还练习不了。

阶段三：Hadoop生态

上面的ZooKeeper和Hive应该放到Hadoop生态中，但是由于这两个框架太重要，太基础，所有放到阶段二中。

该阶段主要包含大数据生产环境下常用的框架

Flume

Flume是一个日志采集传输工具，将服务器日志，落入到大数据的其他组件，方便后期处理。当然它本身也可以对数据做轻微的ETL处理，但是功能定位是传输，所有一般不建议做ETL。

需要掌握的点：

- Flume的内部组成，数据传输过程，拓扑结构
- Flume任务脚本的编写（一般直接从官网获取到sample脚本，然后根据实际情况改写）
- 使用java自定义Interceptor、Source、Sink等功能模块
- Flume的监控工具了解和使用：Ganglia

Kafka

Kafka是一个分布式、分区、多副本、多订阅者，基于ZooKeeper协调的分布式日志系统。是大数据领域中使用最多，最重要的消息队列组件。用于传输和存储短期数据。可以同时支持离线和实时处理，所有经常用于离线的消息传输通道，实时的上层实时存储，有些公司甚至直接使用Kafka作为实时数仓的数据库（因为有副本机制保证数据存储的可靠性）。

需要掌握的点：

- Kafka的组成架构及原理，分区和副本同步机制，存储的数据结构（topic），消费订阅。
- Kafka的安装使用，对接其他组件的常用api，常用的配置参数等。
- Kafka如何保证消费的有序性。如何实现at least once的语义等。

HBase

在大数据之前，数据存储在MySQL这种关系型数据库中，但是受单机的影响，无法存储大量数据。而Hadoop的出现，让大数据的处理（MR）和存储（hdfs）成为可能。但是hdfs只能支持顺序读写数据，访问数据需要遍历整个数据集，同时无法更改历史数据。而谷歌通过BigTable论文，给出了该问题的解决方案。HBase则是该论文的开源实现框架。它本身的实际数据还是依托MR做计算、zk做服务协调、hdfs做存储，但是它通过自身独有特性，相比hdfs，HBase支持随机访问，并巧妙地使用版本的思想，实现了历史数据的更新。

由于HBase是谷歌三大论文之一的实现，建议有时间可以阅读《HBase原理与实践》

需要掌握的点：

- 理解关系型数据库和非关系型数据库的区别，理解什么是结构化、半结构化、非结构化数据。
- 理解传统的行式存储和HBase中列式存储的区别。列式存储为什么可以实现数据压缩和快速读取。
- 掌握HBase的组成架构、存储架构、读写机制、主从机制、RegionServer工作机制等。
- 掌握HBase中的数据结构：Rowkey、Colume、ColumeFamily、Cell、TimeStamp等概念。
- 掌握HBase的安装即使用，和ZooKeeper的关系、Shell命令操作、api操作。
- 理解使用HBase的过滤器，过滤器的分类
- 掌握HBase搭配phoenix后，使用类sql语句操作。
- 掌握HBase建表的属性设置，表设计的原则，Rowkey设计的原则，二级索引的设计。
- 理解HBase的协处理器，及其加载方式。
- （如有精力）理解HBase底层数据的组织方式LSM树，可以使用java实现一个LSM树。

Sqoop

sqoop是传统数据库和大数据数据库的桥梁，可以实现MySQL和Hive的数据互通。底层依赖的是MR计算处理。

需要掌握的点：

- sqoop的原理，导入和导出数据的脚本编写，异常问题的参数配置处理。

Azkaban

前面都是大数据各个方向的组件，而Azkaban则是将每个操作组件的任务进行自动化的调度，将一个大的处理任务里面细分job进行组织的工具。

- 只需要学会怎样将shell脚本、jar包等任务编写成job脚本，并使用Azkaban进行调度配置即可。

OLAP相关

OLAP是在线分析处理，相对于的OLTP在线事务处理。主要用在即席查询，和BI分析领域。这部分设计的框架有很多，如果有其他框架学习完后，可以学习，每个框架的学习时间不长，每天学习2-3小时左右的话，3-4天左右可以掌握一个框架，当然深入到每个框架的设计都有很多知识。但同样的前期建议学习4天左右即可，如果实际使用到再深入了解也可。

涉及的框架：

- Kylin
- Presto
- Impala
- Druid
- ClickHouse
- ElasticSearch

项目练习一：离线数仓

对前面学的所有组件做一个综合的练习，要求较高，坚持做完才算掌握阶段三。

阶段四：批计算准实时框架Spark

Scala

在学习Spark前，需要先学习一门新的编程语言scala。因为Spark的源码是java和scala进行编写。同时spark和后面的实时框架flink，都原生地支持通过scala作为数据处理的语言。

scala其实是在java的基础上封装的，一门完全面向对象、面向函数式编程语言，在大数据领域较常用，相比java更简洁。当然，由于scala是封装了java，spark和flink除了支持scala，也支持java。所以如果只会java，也没有关系，但是掌握scala可以更深入源码。反过来，scala封装在java之上，所以掌握java，学习scala会更容易理解。

推荐书籍：《快学scala》

需要掌握的点：

和掌握一门语言的基础使用一样：变量/运算符、语法、流程控制等，同时scala自有的特性：

- 最重要的是scala的常用算子api，能使用算子实现复杂的数据处理。
- 模式匹配
- 高级函数特性

Spark

Spark是在MapReduce后，大数据的第二代计算引擎。在离线批处理和准实时处理中使用最广。它不仅融合了MR的计算思想，还引入了DAG有向无环图的任务调度思想。同时还有很多先进的思想，如堆外内存等，相比MR，Spark的计算速度提高了近100倍。它由下面几大模块组成：

推荐书籍：《图解Spark核心技术与案例实战》

SparkCore

SparkCore是spark的核心，是Spark其他模块的基础。

需要掌握的点：

- 安装Spark后，运行简单的WordCount案例，然后理解Spark的几种运行模式。
- 理解Spark的数据存储结构RDD的含义，掌握RDD的分区概念，并能自定义分区。
- 理解Spark job的划分，RDD的血缘关系，集群中函数及变量的传递注意事项，RDD的缓存及持久化等。
- 掌握读取不同文件数据，RDD的transform及action算子使用，数据处理后写出。
- 掌握RDD编程进阶：系统累积器、自定义累积器、广播变量等高级使用。

学完SparkCore后，实现部分小项目的数据处理。

SparkSQL

SparkSQL是为了降低spark的使用门槛，以及提高开发效率，在不使用spark的算子情况下，使用SQL语法对处理数据。其实底层还是封装了SparkCore的常用算子。

需要掌握的点：

- 掌握SparkSQL读取不同的文件数据，并处理。

- 掌握DataFrame、DataSet、RDD几种数据存储集合的特性、区别、关系及互相转换。
- 掌握在SparkSQL中自定义函数udf 及 udaf等。
- 理解SparkSQL的SQL语法风格和DSL语法风格的互相转换。（DSL语法用得很少，但某些公司、场景下会用，其实可以完全被SQL语法风格替换。）

SparkStreaming

SparkStreaming是spark的准实时处理模块，它是通过微批处理的方式达到类实时的效果。

需要掌握的点：

- 实现一个实时处理的Demo，掌握Dsteam的创建，对接Kafka数据源的api参数等。
- 掌握实时中的有无状态计算概念，以及实时中的窗口概念。

其他模块（了解）

Spark还有SparkML和SparkGraphX两个模块，分别用于机器学习和图计算，如需要使用到再了解也ok。

Spark内核（重点）

学了上面的Spark的几个模块后，为了更加深入理解Spark，还需要学习部分内核知识及源码，包含架构设计，通信机制、内存管理、任务流程等。

需要掌握的点：

- Spark多种部署模式的区别
- Spark通讯机制和架构演变
- Spark任务提交的源码流程
- Spark的shuffle机制
- Spark的内存管理机制，堆内和堆外内存的规划、分配、和作用。
- 其他...

Spark调优（重点）

这部分是spark在实战中，遇到问题时的处理方案，以及spark编程中需要注意的点，如何实现资源的合理使用，计算效率的提升等。

需要掌握的点：

- 常见的几种调优方案
- 常见的数据倾斜处理方案
- 常见的故障排除方案。
- 常见的内存管理、运行资源调优方案。

项目练习二：Spark实时数仓项目

阶段五：实时计算框架Flink

SparkStreaming是通过微批的方式，实现实时，并不是真正的实时，而Flink则是一个完全实时的计算框架，它是完全基于实时计算理论构建的一个框架。被阿里收购后，在国内这两年越来越火，非常普及，而且在Flink1.1版本之后，致力于实现批流一体，所以未来在批计算方面，很可能也会取代Spark。

推荐书籍《Flink内核原理与实现》

同样Flink也分了几个模块

需要掌握的点：

- 安装部署Flink后，使用Flink分别通过批处理及流处理的方式实现一个WordCount案例，初步感知批流处理的不同。
- 掌握Flink的架构：设计的组件、调度的原理，slots的分配及和并行度的关系，任务链等相关概念。
- 掌握Flink的环境配置，读取数据的source-api，transform-api，写数据的sink-api
- 掌握Flink支持的数据类型
- 掌握Flink的时间语义，windows-api，窗口概念，水位线和窗口的关系，窗口的分类，迟到数据处理。
- 掌握Flink的ProcessFunction-api，Timeserver和定时器的定义等。
- 掌握Flink的状态编程和容错机制，检查点的概念，和实现状态一致性的机制原理等，状态保存等。
- 掌握Flink的Table Api，FlinkSQL下表的定义，动态表的概念，实时数据如何实现更新等。
- 掌握FlinkCEP处理复杂事务。

项目练习三：Flink实时数仓项目