• 考试题型分为简答题，程序分析题， 算法设计题，和综合题 以下内容为课程的每章节中需要理解并掌握的知识点。

第一章 大数据概述

大数据的四个特点：数据量大，数据类型繁多，处理速度快和价值密度低。 大数据对科学研究的影响：

第一种范式：实验科学， 第二种范式：理论科学，第三种范式：计算科学，第 四种范式：数据密集型科学。

云计算的关键技术包括虚拟化，分布式存储，分布式计算和多租户等。 大数据与云计算，物联网的关系是什么？（课本 27 页）

第二章 大数据处理架构 Hadoop

Hadoop 生态系统的主要功能组件包括 Zookeeper, HBase, Hive, Pig, Mahout, Sqoop, Flume, Ambari，HDFS, MapReduce ， Avro 等，每个组件的具体功能是什么？ （课 本 31-34 页）

Hadoop 伪分布式的安装过程， core-site.xml 和 hdfs-site.xml 文件的配置内容。

第三章 分布式文件系统 HDFS

分布式文件系统的结构， HDFS 的相关概念： 包括块， 名称节点， 数据节点， 第 二名称节点。名称节点和数据节点的作用。

第二名称节点解决了什么问题，它的作用是什么？

HDFS 1.0 体系结构的局限性有哪些？

※HDFS 的存储原理

※HDFS 的冗余数据保存策略是什么？

※答： HDFS 默认的冗余复制因子是 3，每个文件块会被同时保存到 3 个地方， 其中， 有两个副本放在同一个机架的不同机器上面， 第 3 个副本放在不同机架的 机器上面， HDFS 副本的放置策略如下：

（1） 如果是在集群内发起写操作请求， 则把第一个副本放置在发起写操作 请求的数据节点上， 实现就近写入数据。如果是在集群外发起写操作 请求， 则从集群内部挑选一台磁盘空间较为充足、CPU 工作不太忙的 数据节点，作为第 1 个副本的存放地。

（2） 第 2 个副本会被放置在与第 1 个副本不同的机架的数据节点上。 （3） 第 3 个副本会被放置在与第 1 个副本相同的机架的其他节点上。

（4） 如果还有更多的副本，则继续从集群中随机选择数据节点进行存放。 ※HDFS 数据复制使用流水线复制的策略，请阐述该策略的细节。

※答： 当客户端要往 HDFS 中写入一个文件时，这个文件会首先被写入本 地， 并被切分成若干个块， 每个块的大小是由 HDFS 的设定值来决定的。每个块 都向 HDFS 集群中的名称节点发起写请求， 名称节点会根据系统中各个数据节

点的使用情况， 选择一个数据节点列表返回给客户端， 然后客户端就把数据首先 写入列表中的第 1 个数据节点， 同时把列表传给第 1 个数据节点， 当第 1 个数据 节点接收到 4KB 数据的时候，写入本地，并且向列表中的第 2 个数据节点发起 连接请求，把自己已经接收到的 4KB 数据和列表传给第 2 个数据节点。当第 2 个数据节点接收到 4KB 数据的时候，写入本地，并且向列表中的第 3 个数据节 点发起连接请求， 依此类推，列表中的多个数据节点形成一条数据复制的流水线， 最后， 当文件写完的时候，数据复制也同时完成。

掌握并熟练使用 HDFS 的编程实践操作

理解 HDFS 的读数据过程和写数据过程，

HDFS 编程实践， ※熟悉常用的 HDFS 操作。

第四章 HBase

（1）HBase 数据模型的相关概念： 表， 行键，列族， 列限定符， 单元格， 时间 戳。概念视图，物理视图，表和 Region ，Region 的定位机制。

（2）HBase 的运行机制，架构， Region 服务器的工作原理， Store 的工作原理， HLog 的工作原理。

※HBase 编程实践

※掌握 HBase 常用的 Shell 命令。会利用 HBase 完成对表的基本操作， 包括添加 数据，浏览数据，创建表等操作。

※在 HBase 中创建一个学生信息表 student, 用来存储学生姓名（姓名为 行键，并且假设姓名不会重复）以及考试成绩， 其中考试成绩是一个列族， 分别存储了各个科目的考试成绩，并向表 student 中添加数据。表的逻辑 视图如下所示。

表 2- 1 学生信息表 student 的表结构

|  |  |  |  |
| --- | --- | --- | --- |
| name | Score | | |
| English | Math | Computer |

表 2-2 需要添加到 student 表中的数据

|  |  |  |  |
| --- | --- | --- | --- |
| name | Score | | |
| English | Math | Computer |
| Mary | 89 | 80 | 90 |
| Mike | 68 | 78 | 79 |

※在操作 Hbase 数据库前，首先需要建立连接，具体代码如下：

Public static void init(){

configuration=HBaseConfiguration.create();

**configuration.set(“hbase.rootdir”,”hdfs://localhost:9000/hbase”);**

try{

connection = ConnectionFactory.createConnection(configuration); admin = connection.getAdmin();

}catch (IOException e) {

e.printStackTrace();

}

}

**※**上述代码中， Configuration 对象用于管理 HBase 的配置信息， 下划线部 分为“hbase.rootdir”这个参数设置具体的值，用于指明HBase 数据库的存储路径， 由于本实例将 HDFS 作为 HBase 的底层存储系统，因此这个参数的值设置为 “hdfs://localhost:9000/hbase”;这步操作配置伪分布式模式下的 HBase。

添加数据操作：

Public static void insertData(String tableName, String rowKey, String colFamily, String col, String val) throws IOException {

Table table = connection.getTable(tableName.valueof(tableName)); put put = new Put(rowKey.getBytes());

put.addColumn(colFamily.getBytes(),col.getBytes(), val.getBytes());

table.put(put);

Table.close();

}

※ 浏览数据操作

编程实践中， 为实现在 Hbase 中的信息表浏览插入的数据， 需执行下列代 码：

※Public static void getData(String **tableName**, String **rowKey**, String

**colFamily**, String **col**)throws IOException{

Table table =connection.getTable(TableName.**valueOf(tableName**));

Get get = new Get (**rowKey.getBytes()**);

get.addColumn(colFamily.getBytes(), col.getBytes());

Result result =**table.get(get)**;

System.out.println(newString(result.getValue(colFamily.getBytes(),

col==null? null:col.getBytes())))

table.close();

}

※要获取姓名为“Mary”的“English”科目的数据，需要指定参数 tableName

为“student” 、rowKey 为“Mary” 、colFamily 为“score” 、col 为“English” . 代码如下：

getData(“student”, “Mary”, “score”, “English”);

※与上述代码等效的 HBase Shell 命令如下所示：

get ‘student’, ‘Mary’,{COLUMN=>‘Score:English’}

第五章 NOSQL 数据库

NoSQL 数据库与关系数据库的比较， NoSQL 的四大类型， 键值数据库， 列族数 据库，文档数据库，图数据库。每种数据库的应用场合和优缺点。NoSQL 的三 个基石 CAP ，BASE 和最终一致性理论。

**※** CAP 理论。 CAP 是指： C：一致性。它是指任何一个读操作总是能够 读到之前完成的写操作的结果， 也就是在分布式环境中，多点的数据是一致的。

A：可用性。它是指快速获取数据，可以在确定的时间内返回操作结果。 P：分区容忍性。

**※**不同产品在设计时是如何运用 CAP 理论的

答： CA。也就是强调一致性（C）和可用性(A)，放弃容忍性(P)。最简单的做 法是把所有与事务相关的内容都放到同一台机器上。这种做法会严重影响系统的 可扩展性。传统的关系数据库（MySQL 、SQLserver 和 PostgreSQL）都采用了这 种设计原则，因此扩展性都比较差。

CP。也就是强调一致性（C）和分区容忍性（P），放弃可用性（A）。当出现 网络分区的情况时， 受影响的服务需要等待数据一致， 因此在等待期间就无法对 外提供服务。Neo4j 、BigTable 和 Hbase 等 NOSQL 数据库都采用了 CP 设计原 则。

AP。也就是强调可用性（A）和分区容忍是(P)，放弃一致性（C）。允许系统 返回不一致的数据。这对于许多 Web2.0 网站是可行的，这些网站的用户首先关 注的是网站服务是否可用。对于 Web2.0 网站而言，可用性和分区容忍性优先级 高于数据一致性， 转而采用最终一致性。DynamoDB、Riak、CouchDB、Cassandra 等 NoSQL 数据库就采用了 AP 设计原则。

BASE 理论，它的基本含义是基本可用，软状态， **※**最终一致性。

基本可用是指一个分布式系统的一部分发生问题变得不可用时， 其他部分仍 然可以正常使用，也就是允许分区失败的情形出现。

软状态是指状态可以有一段时间不同步，具有一定的滞后性。

**※**最终一致性： 一致性的类型包括强一致性和弱一致性， 二者的主要区别在 于在高并发的数据访问操作下， 后续操作是否能够获取最新的数据。最终一致性 是弱一致性的一种特例， 允许后续的访问可以暂时读不到更新后的数据， 但是经 过一段时间之后，用户必须读到更新后的数据。

什么是 NEWSQL 数据库?

第六章 云数据库

**※**云数据库的概念

※云数据库是部署在云计算环境中的虚拟化数据库。

云数据库的特性及与其他数据库的关系。

第七章 MapReduce

MapReduce 模型特点：

在 MapReduce 中， 一个存储在分布式文件系统中的大规模数据集会被切分 成许多独立的小数据块， 这些小数据块可以被多个 Map 任务并行处理。 MapReduce 框架会为每个 Map 任务输入一个数据子集， Map 任务生成的结果会 继续作为 Reduce 任务的输入， 最终由 Reduce 任务输出最后结果， 并写入分布式 文件系统。适合用 MapReduce 来处理的数据集需要满足一个前提条件：待处理 的数据集可以分解成许多个小的数据集， 而且每一个小数据集都可以完全并行地 进行处理。

Map 函数和 Reduce 函数的输入和输出格式，

Map: 输入： <k1, v1>,输出 List（<k2, v2>）将小数据集进一步解析成一批 <key,value>对，输入 Map 函数中进行处理。

Reduce：输入<k2,List(v2)>,输出<k3, v3> 输出的中间结果< k2,List(v2)>中的 List(v2)表示是一批属于同一个 k2 的 Value。

Shuffle 过程详解

Map 端的 Shuffle 过程，Reduce 端的 Shuffle 过程。 （具体过程内容见课本第 135- 138 页）

**※**Map 端的 Shuffle 过程：（1）输入数据和执行 Map 任务， （2）写入缓存 （3）溢写（分区，排序和合并）（4）文件归并

**※**Reduce 端的 Shuffle 过程，（1）领取数据 （2）归并数据 （3）把数据 输入给 Reduce 任务。

**※**为何采用 Combiner 可以减少数据传输量。是否所有的 MapReduce 程序都 可以采用 Combiner？为什么？

**※**答： 因为 Combiner 函数执行合并操作， 可以将那些具有相同 key 的<key, value>的 value 加起来。通过合并操作可以减少键值对的数量。因为 Combiner 操 作可以在 Map 端进行， 可以减少 Map 端传输到 reduce 端的数据传输量。不过并 非所有 MapReduce 程序都可以采用 Combiner，因为 Combiner 的输出是 Reduce 任务的输入， Combiner 绝不可能改变 Reduce 任务最终的计算结果。 一般而言，