范围

第一章 大数据概述

第二章 大数据处理架构 Hadoop

第三章 分布式文件系统 HDFS

第四章 分布式数据库 HBase

第七章 MapReduce

第十章 Spark

题型

七个简答题（第一章、第二章、第三章、第三章、第四章、第七章、第十章）

两个分析题（第一章、第七章）

一个讨论题（开放性）

第一章 大数据概述

大数据的概念（大数据的特点）：

- 数据量大：随着 Web 技术的发展、物联网的推广等因素，数据量呈现爆炸式增长
- 数据类型繁多：总体上可分为结构化数据和非结构化数据
- 处理速度快：需要基于快速生成的数据给出及时的处理与响应
- 价值密度低：许多有价值的数据都是分散在海量数据中

数据产生方式变革的三个阶段：（分析题）

- 运营式系统阶段：数据库的出现使得数据管理的复杂度大大降低，在这个阶段，数据的产生方式是被动的，只有当实际的业务发生时，才会产生新的记录并存入数据库
- 用户原创内容阶段：互联网的出现使得数据传播更加快捷，随着 Web 2.0 的到来和智能终端的普及，大量上网用户成为了内容的生产者，数据量开始急剧增长
- 感知式系统阶段：物联网的发展最终导致了人类社会数据量的第三次跃升，物联网中的大量设备每时每刻都在自动产生大量数据

信息技术发展史上的 3 次信息化浪潮及具体内容：（分析题）

- 第一次浪潮：1980 年前后，以 PC 为标志，解决信息处理问题
- 第二次浪潮：1995 年前后，以 Internet 为标志，解决信息传输问题
- 第三次浪潮：2010 年前后，以物联网、云计算、大数据为标志，解决信息爆炸问题

大数据的影响：

对思维方式的影响：全样而非抽样、效率而非精确、相关而非因果；对社会发展的影响；对就业方面的影响；对科学发展的影响；

科学研究经历了哪四个阶段：

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）

大数据计算模式：

批处理计算（针对大规模数据的批量处理）、流计算（针对流数据的实时计算）、图计算（针对大规模图结构数据的处理）、查询分析计算（大规模数据的存储管理和查询分析）

大数据、云计算、物联网三者之间的区别和联系：

区别：

大数据侧重于对海量数据的存储、处理和分析；

云计算侧重于整合和优化各种 IT 资源，并通过网络的方式提供廉价的、可伸缩的分布式计算能力；

物联网侧重于实现万物互联，应用创新是物联网发展的核心；

联系：

云计算为大数据提供了技术基础，大数据为云计算提供了用武之地；

物联网是大数据的重要来源，大数据为物联网数据分析提供了支持；

云计算为物联网提供了海量的数据存储，物联网为云计算提供了广阔的应用空间；

第二章 大数据处理架构 Hadoop

概念 & 配置 & 生态系统

Hadoop 概念：

Hadoop 是一个开源的、可运行于大规模集群上的分布式计算平台，核心是 Hadoop 分布式文件系统（HDFS）和 MapReduce

Hadoop 的特性：

高可靠：采用冗余数据存储方式，一个副本发生故障时，其他副本也能确保正常运行；

高效：采用分布式存储和分布式处理两大核心技术；

高可扩展、成本低：可以运行在廉价的计算机集群上；

高容错：采用冗余数据存储方式，能够自动保存数据的多个副本；

运行在 Linux 系统中：基于 Java 开发；

支持多种编程语言：应用程序可以使用其他语言编写；

Hadoop 生态系统以及每个部分的具体功能：

HDFS：分布式文件系统，具有处理超大数据、流式处理、可以运行在廉价商用服务器上等优点；

HBase：非关系型的分布式数据库，是一个高可靠、高性能、可伸缩、实时读写、分布式的列式数据库；

MapReduce：分布式计算框架，用于大规模数据集的并行计算，将复杂的、运行于大规模集群上的并行计算过程高度抽象为 Map 和 Reduce 两个函数，允许用户在不了解分布式系统底层细节的情况下开发并行应用程序；

Hive：基于 Hadoop 的数据仓库工具，可以用于对 Hadoop 文件中的数据集进行数据整理、特殊查询和分析存储；

Pig：数据流语言和运行环境，适合于使用 Hadoop 和 MapReduce 平台来查询大型半结构化数据集；

YARN：资源管理和调度器；

Sqoop：用于在 Hadoop 和传统数据库之间进行数据传递；

ZooKeeper：提供分布式协调一致性服务；

第三章 分布式文件系统 HDFS

概念 & 体系结构 & 读写过程

分布式文件系统在物理结构上是由计算机集群中的多个节点构成的，这些节点分为两类：一类叫主节点，或者被称为名称节点；一类叫从节点，或者被称为数据节点

在存储时，由名称节点分配存储位置，然后由客户端把数据直接写入相应数据节点；在读取时，客户端从名称节点获得数据节点和文件块的映射关系，然后就可以到相应位置访问文件块；数据节点也要根据名称节点的命令创建、删除和复制数据块

HDFS 的一些概念：

- 块：为了提高磁盘的读写效率，文件按一定的大小被分为若干个数据块，块是数据读写操作的最小单位
- 名称节点：负责文件和目录的创建、删除和重命名等，同时管理着数据节点和文件块的映射关系
- 数据节点：负责数据的存储和读取

HDFS 中的块和普通文件系统块的区别：

- 在传统的文件系统中，为了提高磁盘的读写效率，一般以数据块为单位
- HDFS 中的块默认大小为 64 MB，HDFS 中的文件按一定的大小被分为若干个数据块，每个块作为独立的单元进行存储

HDFS 中的名称节点和数据节点的具体功能：

- 名称节点： 名称节点保存了所有的元数据信息，其中最核心的两大文件是 FsImage 和 EditLog，名称节点负责文件和目录的创建、删除和重命名等，同时管理着数据节点和文件块的映射关系
- 数据节点：是 HDFS 的工作节点，负责数据的存储和读取

为什么要采用第二名称节点：

当名称节点重启时，需要将 FsImage 加载到内存中，然后逐条执行 EditLog 中的记录来更新 FsImage，当 EditLog 过大时，会导致整个过程非常缓慢。为了解决这个问题，引入了第二名称节点，用来协助名称节点，加速名称节点的启动

第二名称节点的功能：

一方面可以完成 EditLog 和 FsImage 的合并操作，减小 EditLog 文件的大小，缩短名称节点重启时间；另一方面可以作为名称节点的检查点，保存名称节点中的元数据信息

HDFS 体系结构的局限性：

- 命名空间的限制：名称节点保存在内存中，其能够容纳对象的个数是有限的；
- 性能的瓶颈：整个分布式文件系统的吞吐量受限于单个名称节点的吞吐量；
- 隔离问题：由于集群中只有一个名称节点，只有一个命名空间，因此无法对不同应用程序进行隔离；
- 集群的可用性：一旦这个唯一的名称节点发生故障，会导致整个集群变得不可用；

HDFS 数据错误与恢复：主要包含以下三种情况

名称节点出错：

- 名称节点保存了所有的元数据信息，其中最核心的两大文件是 FsImage 和 EditLog，如果这两个文件发生损坏，那么整个 HDFS 实例将失效。Hadoop 采用两种机制来确保名称节点的安全，一是把名称节点上的元数据信息同步存储到其他文件系统中，比如远程挂载的网络文件系统；二是运行一个第二名称节点，当名称节点死机后，可以利用第二名称节点中的元数据信息进行系统恢复，但这样仍会丢失部分数据

数据节点出错：

- 每个数据节点都会定期向名称节点发送心跳信息，向名称节点报告自己的状态。当数据节点或网络发生故障时，名称节点就会因为无法收到这些数据节点的信息而将其标记为死机。这时，有可能出现一种情形，即由于一些数据节点的不可用，会导致一些数据块的副本数量少于冗余因子，名称节点会定期检查这种情况，一旦发现，就会启动数据冗余复制，为它生成新的副本

数据出错：

- 网络传输和磁盘错误等因素都会导致数据出错。客户端在每次读取数据的时候都会采用哈希函数计算所读取数据块的消息摘要，并与文件创建时计算生成的消息摘要做对比，若校验不匹配，则客户端会请求到另外一个数据节点读取该文件块，并且向名称节点报告这个文件块有错误，名称节点会定期检查并重新复制出错的块

HDFS 读数据的七个过程：

1. 客户端调用 open() 方法，打开文件
2. 从名称节点获取数据块的信息
3. 客户端调用 read() 方法，与最近的数据节点建立连接并读取数据
4. 从该数据节点读取数据到客户端
5. 从名称节点获取下一个数据块的信息（可能发生）
6. 从该数据节点读取数据到客户端
7. 客户端调用 close() 方法，关闭文件

HDFS 写数据的七个过程：

1. 客户端调用 create() 方法，创建文件
2. 在名称节点中创建文件元数据
3. 客户端调用 write() 方法，向文件中写入数据
4. 写好的数据被分包，最终被打包成数据包，写入各个数据节点
5. 接收各个数据节点返回的确认包
6. 客户端调用 close() 方法，关闭文件
7. 写操作完成

第四章 分布式数据库 HBase

数据模型：行键、列族、列限定符、时间戳 & 实现原理 & 运行机制

特点、优势：

HBase 是一个高可靠、高性能、面向列、可伸缩的分布式数据库，主要用于存储半结构化和非结构化的松散数据

描述 HBase 数据模型：

- 行键：用来唯一标识 HBase 表中的每一行，可以是任意字符串
- 列族：是基本的访问控制单元，需要在创建表的时候就定义好
- 列限定符：列族中的数据通过列限定符来定位，不需要事先定义
- 时间戳：每个单元格都保存着同一份数据的多个版本，使用时间戳进行索引

HBase 的功能组件及作用：

HBase 的实现包括 3 个主要的功能组件：库函数、一个 Master 主服务器、许多个 Region 服务器，库函数链接到每个客户端，Master 主服务器负责管理和维护 HBase 表的分区信息，比如一个表被分成了哪些 Region，每个 Region 被存放在哪台 Region 服务器上，同时也负责维护 Region 服务器列表，此外还处理模式变化，如表和列族的创建，Region 服务器负责存储和维护分配给自己的 Region，处理来自客户端的读写请求

Region 的概念：

在一个 HBase 中，存储了许多用户数据表，对于每一个表而言，表中包含的行的数量可能非常大，无法存储在一台机器上，因此需要根据行键的值对表中的行进行分区，每个行区间构成一个分区，被称为 Region ，Region 包含了位于某个值域区间内的所有数据，是负载均衡和数据分发的基本单位，Master 主服务器会把不同的 Region 分配到不同的 Region 服务器上，每个 Region 服务器负责管理一个 Region 集合

每个 Region 都有一个 RegionID 来标识它的唯一性，一个 Region 标识符可以表示为 表名 + 开始主键 + RegionID

HBase 的三层结构：

为了定位用户数据表每个 Region 所在的位置，可以构建一张映射表来表示 Region 和 Region 服务器之间的对应关系，表中的每行包含两项内容：Region 标识符、Region 服务器标识，这个映射表包含了关于 Region 的元数据，因此被称为元数据表，也叫 .META. 表

当一个 HBase 表中的 Region 数量非常大的时候，.META. 表的行数也会非常多，一个服务器保存不下，也需要分区存储到不同的服务器上，因此，.META. 表也会被分为许多个 Region ，为了定位这些 Region ，就需要构建一个新的映射表，这个新的映射表就是根数据表，也叫 -ROOT- 表

- 第一层：ZooKeeper 文件，用于记录 -ROOT- 表的位置信息
- 第二层： -ROOT- 表，用于记录 .META. 表 Region 的位置信息，-ROOT- 表不能再被分区，永远只存放在一个 Region 中，通过 -ROOT- 表可以访问 .META. 表中的数据
- 第三层： .META. 表，用于记录用户数据表 Region 的位置信息，.META. 表可以被分为许多个 Region，保存了 HBase 中所有用户数据表 Region 的位置信息

对应三层结构的三级寻址过程：

客户端访问用户数据之前，首先需要访问 ZooKeeper，获取 -ROOT- 表的位置信息，然后访问 -ROOT- 表，获得 .META. 表的信息，接着访问 .META. 表，找到所需的 Region 具体位于哪个 Region 服务器，最后才会到该 Region 服务器读取数据

第七章 MapReduce

概念 & 工作流程 & 分布式编程：以 WordCount 为例

MapReduce 的概念：

MapReduce 是一种分布式计算框架，用于大规模数据集的并行计算，将复杂的、运行于大规模集群上的并行计算过程高度抽象为 Map 和 Reduce 两个函数

计算向数据靠拢的原因：

移动大规模数据需要大量的网络传输开销，出于这个因素，在一个集群中，只要有可能，MapReduce 框架就会将 Map 程序就近地在 HDFS 数据所在的节点运行，即将计算节点与存储节点放在一起运行，进而减少节点间的移动数据开销

JobTracker 的功能、作用：

JobTracker 在 Map 端的 Shuffle 过程中会一直监测 Map 任务的执行，当监测到一个 Map 任务完成时，会立即通知相关的 Reduce 任务来领取数据，然后开始 Reduce 端的 Shuffle 过程

MapReduce 工作流程：

核心思想是分而治之，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为许多个独立的小数据集，这些小数据集可以被多个 Map 任务在多台机器上并行处理，当 Map 任务结束后，会生成许多键值对形式的中间结果，这些结果会被分发到多个 Reduce 任务在多台机器上并行处理，具有相同 key 的键值对会被发送到同一个 Reduce 任务，Reduce 任务会对中间结果进行汇总处理得到最后结果，并写入分布式文件系统

MapReduce 执行阶段（题目里面的工作流程？）：

1. 使用 InputFormat 对输入进行预处理，然后将输入文件在逻辑上分为多个 InputSplit
2. 使用 RecordReader 根据 InputSplit 中的信息处理具体记录，加载数据并转换为键值对，输入给 Map 任务
3. Map 任务根据用户自定义的逻辑，生成许多键值对形式的中间结果
4. 通过 shuffle 过程对这些中间结果进行处理，包括分区、排序、可能的合并和归并操作，使得 Reduce 任务能够并行处理
5. Reduce 任务根据用户自定义的逻辑，对中间结果进行汇总处理得到最后结果，输出给 OutputFormat
6. OutputFormat 会验证输出目录和输出结果类型，将结果写入分布式文件系统

Shuffle 过程：

Shuffle 过程分为 Map 端的过程和 Reduce 端的过程

Map 端的过程包括：

1. 输入数据和执行 Map 任务
2. 将输出结果写入缓存
3. 缓存满时启动溢写操作，在写入磁盘之前还要进行分区、排序以及可能的合并操作
4. 将所有溢写文件中的数据归并为一个大文件，具有相同 key 的键值对会被归并成一个新的键值对

Reduce 端的过程包括：

1. 当收到 JobTracker 的通知后，到相关的 Map 任务所在机器上领取属于自己处理的数据
2. 领取的数据会被存放在缓存中，缓存满时会启动溢写，当数据全部领取完后会进行归并操作
3. 将经过归并后得到的若干个大文件输入给 Reduce 任务

MapReduce 具体实例的执行过程、算法思路：（分析题）

词频统计：

文件合并和去重：

算法思路：在 Shuffle 过程中，具有相同 key 的键值对会被归并为一个新的键值对

执行过程：在 Map 任务中，将输入文件中每一行的内容作为 key，以空字符串作为 value。经过 Shuffle 过程，相同 key 的键值对会被归并，即重复的记录会被合并。然后将这些中间数据分发给 Reduce 任务，在 Reduce 任务中，将每个键值对的 key 输出到分布式文件系统即可

对输入文件的排序：

算法思路：在 Shuffle 过程中，会自动根据 key 值对输出的键值对进行排序

执行过程：在 Map 任务中，将读入的数据转化为 IntWritable 型，然后作为 key 值输出，value 可以任意设置。MapReduce 会按照 key 值的大小对键值对进行排序，相同 key 的键值对会被归并。在 Reduce 任务中，将一个从 1 开始递增的整数作为 key 表示排序次数，将输入的 key 作为 value ，将输入的 value-list 中的元素个数作为输出次数，输出到分布式文件系统即可

对给定表格进行信息挖掘：

算法思路：在 Shuffle 过程中，具有相同 key 的键值对会被归并为一个新的键值对

执行过程：在 Map 任务中，将输入文件按照空格分割成 child 和 parent，对于文件的每一行，输出两个键值对，这两个键值对分别以 parent 和 child 作为 key，在输入的每一行中添加区分标记作为它们的 value。经过 Shuffle 过程，相同 key 的键值对会被归并。在 Reduce 任务中，接受到的键值对中的 value-list 中即包含了祖孙关系

第十章 Spark

提出的原因 & 生态系统

Spark 特点：

- 运行速度快：使用 DGA 执行引擎，支持循环数据流与内存计算
- 容易使用：支持多种语言编程
- 通用性：提供了完整而强大的技术栈
- 运行模式多样：可运行于独立的集群模式、Hadoop 以及各种云环境

Spark 和 Hadoop 的比较：

Hadoop 存在以下的一些缺点：

表达能力有限、磁盘 IO 开销大、延迟高

相比于 Hadoop MapReduce ，Spark 主要具有如下优点：

- 不局限于 Map 和 Reduce 操作，提供了多种数据集操作类型，编程模型更加灵活
- 提供了内存计算，可将中间结果放到内存中，对于迭代运算效率更高
- 基于 DAG 的任务调度执行机制要优于Hadoop MapReduce 的迭代执行机制