**第一章**

1.试述信息技术发展史上的3次信息化浪潮及具体内容。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信息化浪潮 | 发生时间 | 标志 | 解决问题 | 代表公司 |
| 第一次浪潮 | 1980年前后 | 个人计算机 | 信息处理 | Intel、AMD、IBM、苹果、微软、联想、戴尔、惠普等 |
| 第二次浪潮 | 1995年前后 | 互联网 | 信息传输 | 雅虎、谷歌、阿里巴巴、百度、腾讯等 |
| 第三次浪潮 | 2010年前后 | 物理网、云计算和大数据 | 信息爆炸 | 将涌现出一批新的市场标杆企业 |

1. 试述数据产生方式经历的几个阶段

答： 运营式系统阶段，用户原创内容阶段，感知式系统阶段。

1. 试述大数据的4个基本特征

答：数据量大、数据类型繁多、处理速度快和价值密度低。

1. 试述大数据时代的“数据爆炸”的特性

答：大数据时代的“数据爆炸”的特性是，人类社会产生的数据一致都以每年50%的速度增长，也就是说，每两年增加一倍。

1. 数据研究经历了哪4个阶段？

答：人类自古以来在科学研究上先后历经了实验、理论、计算、和数据四种范式。

1. 试述大数据对思维方式的重要影响

答：大数据时代对思维方式的重要影响是三种思维的转变：全样而非抽样，效率而非精确，相关而非因果。

1. 大数据决策与传统的基于数据仓库的决策有什么区别

答：数据仓库具备批量和周期性的数据加载以及数据变化的实时探测、传播和加载能力，能结合历史数据和实时数据实现查询分析和自动规则触发，从而提供对战略决策和战术决策。

大数据决策可以面向类型繁多的、非结构化的海量数据进行决策分析。

1. 举例说明大数据的基本应用

答：

|  |  |
| --- | --- |
| 领域 | 大数据的应用 |
| 金融行业 | 大数据在高频交易、社区情绪分析和信贷风险分析三大金融创新领域发挥重要作用。 |
| 汽车行业 | 利用大数据和物联网技术的五人驾驶汽车，在不远的未来将走进我们的日常生活 |
| 互联网行业 | 借助于大数据技术，可以分析客户行为，进行商品推荐和有针对性广告投放 |
| 个人生活 | 大数据还可以应用于个人生活，利用与每个人相关联的“个人大数据”，分析个人生活行为习惯，为其提供更加周全的个性化服务。 |

1. 举例说明大数据的关键技术

答：批处理计算，流计算，图计算，查询分析计算

1. 大数据产业包含哪些关键技术。

答：IT基础设施层、数据源层、数据管理层、数据分析层、数据平台层、数据应用层。

1. 定义并解释以下术语：云计算、物联网

答： 云计算：云计算就是实现了通过网络提供可伸缩的、廉价的分布式计算机能力，用户只需要在具备网络接入条件的地方，就可以随时随地获得所需的各种IT资源。

物联网是物物相连的互联网，是互联网的延伸，它利用局部网络或互联网等通信技术把传感器、控制器、机器、人类和物等通过新的方式连在一起，形成人与物、物与物相连，实现信息化和远程管理控制。

1. 详细阐述大数据、云计算和物联网三者之间的区别与联系。

|  |  |
| --- | --- |
| 大数据、云计算和物联网的区别 | 大数据、云计算和物联网的联系 |
| 大数据侧重于海量数据的存储、处理与分析，海量数据中发现价值，服务于生产和生活；云计算本质上皆在整合和优化各种IT资源并通过网络已服务的方法，廉价地提供给用户；物联网的发展目标是实现呜呜向量，应用创新是物联网的核心 | 从整体来看，大数据、云计算和物联网这三者是相辅相成的。大数据根植于云计算，大数据分析的很多技术都来自于云计算，云计算的分布式存储和管理系统提供了海量数据的存储和管理能力，没有这些云计算技术作为支撑，大数据分析就无从谈起。物联网的传感器源源不断的产生大量数据，构成了大数据的重要数据来源，物联网需要借助于云计算和大数据技术，实现物联网大数据的存储、分析和处理。 |

**第二章**

1. 试述hadoop和谷歌的mapreduce、gfs等技术之间的关系

答：Hadoop的核心是分布式文件系统HDFS和MapReduce，HDFS是谷歌文件系统GFS的开源实现，MapReduces是针对谷歌MapReduce的开源实现。

1. 试述Hadoop具有哪些特性。

答：高可靠性，高效性，高可扩展性，高容错性，成本低，运行在Linux平台，支持多种编程语言

1. 试述Hadoop在各个领域的应用情况。

答：2007年，雅虎在Sunnyvale总部建立了M45——一个包含了4000个处理器和1.5PB容量的Hadooop集群系统；

Facebook主要将Hadoop平台用于日志处理，推荐系统和数据仓库等方面；

百度主要使用Hadoop于日志的存储和统计、网页数据的分析和挖掘、商业分析、在线数据反馈、网页聚类等。

1. 试述Hadoop的项目结构以及每个部分的具体功能。

答：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pig** | **Chukwa** | | **Hive** | | **HBase** |
| **MapReduce** | | **HDFS** | | **Zookeeper** | |
| **Common** | | | **Avro** | | |

Commeon是为Hadoop其他子项目提供支持的常用工具，主要包括文件系统、RPC和串行化库

Avro是为Hadoop的子项目，用于数据序列化的系统，提供了丰富的数据结构类型、快速可压缩的二进制数据格式、存储持续性数据的文件集、远程调用的功能和简单的动态语言集成功能。

HDFS是Hadoop项目的两个核心之一，它是针对谷歌文件系统的开源实现。

HBase是一个提高可靠性、高性能、可伸缩、实时读写、分布式的列式数据库，一般采用HDFS作为其底层数据存储。

MapReduce是针对谷歌MapReduce的开源实现，用于大规模数据集的并行运算。

Zoookepper是针对谷歌Chubby的一个开源实现，是高效和可靠的协同工作系统，提供分布式锁之类的基本服务，用于构建分布式应用，减轻分布式应用程序所承担的协调任务。

Hive是一个基于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、特殊查询和分布存储。

Pig是一种数据流语言和运行环境，适合于使用Hadoop和MapReducce平台上查询大型半结构化数据集。

Sqoop可以改进数据的互操作性，主要用来在H大哦哦哦配合关系数据库之间交换数据。

Chukwa是一个开源的、用于监控大型分布式系统的数据收集系统，可以将各种类型的数据收集成适合Hadoop处理的文件，并保存在HDFS中供Hadoop进行各种 MapReduce操作。

**第三章**

1. 试述分布式文件系统设计的需求。

|  |  |  |
| --- | --- | --- |
| 设计需求 | 含义 | HDFS的实现情况 |
| 透明性 | 具备访问透明性、位置透明性、性能、和伸缩透明性 | 只能提供一定程度的访问透明性，完全支持位置透明性、性能和伸缩透明性 |
| 并发控制 | 客户端对于文件的读写不应该影响其他客户端对同一个文件的读写 | 机制非常简单，任何时候都只允许有一个程序写入某个文件 |
| 文件复制 | 一个文件可以拥有不同位置的多个副本 | HDFS采用了多副本机制 |
| 硬件和操作系统的异构性 | 可以在不同的操作系统和计算机上实现同样的客户端和服务端程序 | 采用Java语言开发，具有很好的跨平台能力 |
| 可伸缩性 | 支持节点的动态加入或退出 | 建立在大规模廉价机器上的分布式文件系统集群，具有很好的伸缩性 |
| 容错 | 保证文件服务在客户端或者服务端出现问题的时候能正常使用 | 具有多副本机制和故障自动检测、恢复机制 |
| 安全 | 保证系统的安全性 | 安全性较弱 |

1. 分布式文件系统是如何实现较高水平扩展的？

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”(Master Node)或者也被称为“名称结点”(NameNode)，另一类叫“从节点”（Slave Node）或者也被称为“数据节点”(DataNode)

1. 试述HDFS中的块和普通文件系统中的块的区别。

答：在传统的文件系统中，为了提高磁盘读写效率，一般以数据块为单位，恶如不是以字节为单位。

HDFS中的块，默认一个块大小为64MB，而HDFS中的文件会被拆分成多个块，每个块作为独立的单元进行存储。HDFS在块的大小的设计上明显要大于普通文件系统。

1. 试述HDFS中的名称节点和数据节点的具体功能。

答：名称节点负责管理分布式文件系统系统的命名空间，记录分布式文件系统中的每个文件中各个块所在的数据节点的位置信息；

数据节点是分布式文件系统HDFS的工作节点，负责数据的存储和读取，会根据客户端或者是名称节点的调度来进行数据的存储和检索，并向名称节点定期发送自己所存储的块的列表。

hadoop fs -ls <path> 显示<path>指定的文件的详细信息

hadoop fs -cat <path> 将<path>指定的文件的内容输出到标准输出

hadoop fs -mkdir <path> 创建<path>指定的文件夹

hadoop fs -get [-ignorecrc] [-crc] <src><localdst> 复制<src>指定的文件到本地文件系统<localdst>指定的文件或文件夹。-ignorecrc选项复制CRC校验失败的文件。使用-crc选项复制文件以及CRC信息。

hadoop fs -put <localsrc><dst> 从本地文件系统中复制<localsrc>指定的单个或多个源文件到<dst>指定的目标文件系统中。也支持从标准输入(stdin)中读取输入写入目标文件系统。

hadoop fs -rmr <path> 删除<path>指定的文件夹及其的所有文件

**第四章**

1. 试述在Hadoop体系架构中HBase与其他组成部分的相互关系。

答： HBase利用Hadoop MapReduce来处理HBase中的海量数据，实现高性能计算；利用Zookeeper作为协同服务，实现稳定服务和失败恢复；使用HDFS作为高可靠的底层存储，利用廉价集群提供海量数据存储能力; Sqoop为HBase的底层数据导入功能，Pig和Hive为HBase提供了高层语言支持，HBase是BigTable的开源实现。

1. 请阐述HBase和BigTable的底层技术的对应关系

答：

|  |  |  |
| --- | --- | --- |
| 项目 | BigTable | HBase |
| 文件存储系统 | GFS | HDFS |
| 海量数据处理 | MapReduce | Hadoop MapReduce |
| 协同服务管理 | Chubby | Zookeeper |

1. 请阐述HBase和传统关系数据库的区别

答：

|  |  |  |
| --- | --- | --- |
| 区别 | 传统关系数据库 | HBase |
| 数据类型 | 关系模型 | 数据模型 |
| 数据操作 | 插入、删除、更新、查询、多表连接 | 插入、查询、删除、清空，无法实现表与表之间关联 |
| 存储模式 | 基于行模式存储，元组或行会被连续地存储在磁盘也中 | 基于列存储，每个列族都由几个文件保存，不同列族的文件是分离的 |
| 数据索引 | 针对不同列构建复杂的多个索引 | 只有一个行键索引 |
| 数据维护 | 用最新的当前值去替换记录中原来的旧值 | 更新操作不会删除数据旧的版本，而是生成一个新的版本 |
| 可伸缩性 | 很难实现横向扩展，纵向扩展的空间也比较有限 | 轻易地通过在集群中增加或者减少硬件数量来实现性能的伸缩 |

1. HBase有哪些类型的访问接口？

答：HBase提供了Native Java API , HBase Shell , Thrift Gateway , REST GateWay , Pig , Hive 等访问接口。

1. 请以实例说明HBase数据模型。

列限定符

列族

答：

|  |  |  |  |
| --- | --- | --- | --- |
|  | Info | | |
| Name | Major | Email |
| 201505001 | Luo Min | Math | Luo@qq.com |
| 201505002 | Liu Jun | Math | liu@qq.com |
| 201505003 | Xie You | Math | xie@qq.com  you@163.com |

行键

单元格

单元格有两个时间戳ts1和ts2

每个时间戳对应一个数据版本

1. 分别解释HBase中行键、列键和时间戳的概念
2. 行键是唯一的，在一个表里只出现一次，否则就是在更新同一行，行键可以是任意的字节数组。
3. 列族需要在创建表的时候就定义好，数量也不宜过多。列族名必须由可打印字符组成，创建表的时候不需要定义好列。
4. 时间戳，默认由系统指定，用户也可以显示设置。使用不同的时间戳来区分不同的版本。
5. 请举个实例来阐述HBase的概念视图和物理视图的不同

HBase数据概念视图

|  |  |  |  |
| --- | --- | --- | --- |
| 行键 | 时间戳 | 列族contents | 列族anchor |
| “com.cnn.www” | T5 |  | Anchor:cnnsi.com=”CNN” |
| T3 |  | Anchor:my.look.ca=”CNN” |
| “com.cnn.www” | T3 | Content:html=”<html>...” |  |
| T2 | Content:html=”<html>...” |  |
| T1 | Content:html=”<html>...” |  |

HBase数据物理视图

|  |  |  |
| --- | --- | --- |
| 行键 | 时间戳 | 列族anchor |
| “com.cnn.www” | T5 | Anchor:cnnsi.com=”CNN” |
| T4 | Anchor:my.look.ca=”CNN” |

|  |  |  |
| --- | --- | --- |
| 行键 | 时间戳 | 列族contents |
| “com.cnn.www” | T3 | Content:html=”<html>...” |
| T2 | Content:html=”<html>...” |
| T1 | Content:html=”<html>...” |

在HBase的概念视图中，一个表可以视为一个稀疏、多维的映射关系。

在物理视图中，一个表会按照属于同一列族的数据保存在一起

1. 试述HBase各功能组建及其作用

（1）库函数：链接到每个客户端；

（2）一个Master主服务器：主服务器Master主要负责表和Region的管理工作；

（3）许多个Region服务器：Region服务器是HBase中最核心的模块，负责维护分配给自己的Region，并响应用户的读写请求

1. 请阐述HBase的数据分区机制。

答： HBase采用分区存储，一个大的表会被分拆许多个Region，这些Region会被分发到不同的服务器上实现分布式存储。

1. HBase中的分区是如何定位的。

通过构建的映射表的每个条目包含两项内容，一个是Regionde 标识符，另一个是Region服务器标识，这个条目就标识Region和Region服务器之间的对应关系，从而就可以知道某个Region被保存在哪个Region服务器中。

1. 试述HBase的三层结构中各层次的名称和作用。

|  |  |  |
| --- | --- | --- |
| **层次** | **名称** | **作用** |
| 第一层 | Zookeeper文件 | 记录了-ROOT-表的位置信息 |
| 第二层 | -ROOT-表 | 记录了.META.表的Region位置信息  -ROOT-表只能有一个Region。通过-ROOT-表，就可以访问.META.表中的数据 |
| 第三层 | .META.表 | 记录了用户数据表的Region位置信息，.META.表可以有多个Region，保存了HBase中所有用户数据表的Region位置信息 |

1. 请阐述HBase的三层结构下，客户端是如何访问到数据的。

答：首先访问Zookeeper，获取-ROOT表的位置信息，然后访问-Root-表，获得.MATA.表的信息，接着访问.MATA.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。

1. 试述HBase系统基本架构以及每个组成部分的作用。

（1）客户端

客户端包含访问HBase的接口，同时在缓存中维护着已经访问过的Region位置信息，用来加快后续数据访问过程

（2）Zookeeper服务器

Zookeeper可以帮助选举出一个Master作为集群的总管，并保证在任何时刻总有唯一一个Master在运行，这就避免了Master的“单点失效”问题

（3）Master

主服务器Master主要负责表和Region的管理工作：管理用户对表的增加、删除、修改、查询等操作；实现不同Region服务器之间的负载均衡；在Region分裂或合并后，负责重新调整Region的分布；对发生故障失效的Region服务器上的Region进行迁移

（4）Region服务器

Region服务器是HBase中最核心的模块，负责维护分配给自己的Region，并响应用户的读写请求

1. 请阐述Region服务器向HDFS文件系统中读写数据的基本原理

Region服务器内部管理一系列Region对象和一个HLog文件，其中，HLog是磁盘上面的记录文件，它记录着所有的更新操作。每个Region对象又是由多个Store组成的，每个Store对象了表中的一个列族的存储。每个Store又包含了MemStore和若干个StoreFile，其中，MemStore是在内存中的缓存。

1. 试述HStore的工作原理

每个Store对应了表中的一个列族的存储。每个Store包括一个MenStore缓存和若干个StoreFile文件。MenStore是排序的内存缓冲区，当用户写入数据时，系统首先把数据放入MenStore缓存，当MemStore缓存满时，就会刷新到磁盘中的一个StoreFile文件中，当单个StoreFile文件大小超过一定阈值时，就会触发文件分裂操作。

1. 试述HLog的工作原理

答：HBase系统为每个Region服务器配置了一个HLog文件，它是一种预写式日志（Write Ahead Log），用户更新数据必须首先写入日志后，才能写入MemStore缓存，并且，直到MemStore缓存内容对应的日志已经写入磁盘，该缓存内容才能被刷写到磁盘。

1. 在HBase中，每个Region服务器维护一个HLog，而不是为每个Region都单独维护一个HLog。请说明这种做法的优缺点。

优点： 多个Region对象的更新操作所发生的日志修改，只需要不断把日志记录追加到单个日志文件中，不需要同时打开、写入到多个日志文件中。

缺点：如果一个Region服务器发生故障，为了恢复其上次的Region对象，需要将Region服务器上的对象，需要将Region服务器上的HLog按照其所属的Region对象进行拆分，然后分发到其他Region服务器上执行恢复操作。

1. 当一台Region服务器意外终止时，Master如何发现这种意外终止情况？为了恢复这台发生意外的Region服务器上的Region,Master应该做出哪些处理(包括如何使用HLog进行恢复)?

Zookeeper会实时监测每个Region服务器的状态，当某个Region服务器发生故障时，Zookeeper会通知Master。

Master首先会处理该故障Region服务器上面遗留的HLog文件，这个遗留的HLog文件中包含了来自多个Region对象的日志记录。

系统会根据每条日志记录所属的Region对象对HLog数据进行拆分，分别放到相应Region对象的目录下，然后，再将失效的Region重新分配到可用的Region服务器中，并把与该Region对象相关的HLog日志记录也发送给相应的Region服务器。

Region服务器领取到分配给自己的Region对象以及与之相关的HLog日志记录以后，会重新做一遍日志记录中的各种操作，把日志记录中的数据写入到MemStore缓存中，然后，刷新到磁盘的StoreFile文件中，完成数据恢复。

**第五章**

1. 如何准确理解NoSQL的含义？

NoSQL是一种不同于关系数据库的数据库管理系统设计方式，是对非关系型数据库的一类统称，它采用的数据模型并非传统关系数据库的关系模型，而是类似键/值、列族、文档等非关系模型。

1. 试述关系数据库在哪些方面无法满族Web2.0应用的需求。

关系数据库已经无法满足Web2.0的需求。主要表现在以下几个方面：

（1）无法满足海量数据的管理需求

（2）无法满足数据高并发的需求

（3）无法满足高可扩展性和高可用性的需求

1. 请比较NoSQL数据库和关系数据库的优缺点。

|  |  |  |  |
| --- | --- | --- | --- |
| 比较标准 | RDBMS | NoSQL | 备注 |
| 数据库原理 | 完全支持 | 部分支持 | RDBMS有关系代数理论作为基础  NoSQL没有统一的理论基础 |
| 数据规模 | 大 | 超大 | RDBMS很难实现横向扩展，纵向扩展的空间也比较有限，性能会随着数据规模的增大而降低  NoSQL可以很容易通过添加更多设备来支持更大规模的数据 |
| 数据库模式 | 固定 | 灵活 | RDBMS需要定义数据库模式，严格遵守数据定义和相关约束条件  NoSQL不存在数据库模式，可以自由灵活定义并存储各种不同类型的数据 |
| 查询效率 | 快 | 可以实现高效的简单查询，但是不具备高度结构化查询等特性，复杂查询的性能不尽人意 | RDBMS借助于索引机制可以实现快速查询（包括记录查询和范围查询）  很多NoSQL数据库没有面向复杂查询的索引，虽然NoSQL可以使用MapReduce来加速查询，但是，在复杂查询方面的性能仍然不如RDBMS |
| 一致性 | 强一致性 | 弱一致性 | RDBMS严格遵守事务ACID模型，可以保证事务强一致性  很多NoSQL数据库放松了对事务ACID四性的要求，而是遵守BASE模型，只能保证最终一致性 |
| 数据完整性 | 容易实现 | 很难实现 | 任何一个RDBMS都可以很容易实现数据完整性，比如通过主键或者非空约束来实现实体完整性，通过主键、外键来实现参照完整性，通过约束或者触发器来实现用户自定义完整性  但是，在NoSQL数据库却无法实现 |
| 扩展性 | 一般 | 好 | RDBMS很难实现横向扩展，纵向扩展的空间也比较有限  NoSQL在设计之初就充分考虑了横向扩展的需求，可以很容易通过添加廉价设备实现扩展 |
| 可用性 | 好 | 很好 | RDBMS在任何时候都以保证数据一致性为优先目标，其次才是优化系统性能，随着数据规模的增大，RDBMS为了保证严格的一致性，只能提供相对较弱的可用性  大多数NoSQL都能提供较高的可用性 |
| 标准化 | 是 | 否 | RDBMS已经标准化（SQL）  NoSQL还没有行业标准，不同的NoSQL数据库都有自己的查询语言，很难规范应用程序接口  StoneBraker认为：NoSQL缺乏统一查询语言，将会拖慢NoSQL发展 |
| 技术支持 | 高 | 低 | RDBMS经过几十年的发展，已经非常成熟，Oracle等大型厂商都可以提供很好的技术支持  NoSQL在技术支持方面仍然处于起步阶段，还不成熟，缺乏有力的技术支持 |
| 可维护性 | 复杂 | 复杂 | RDBMS需要专门的数据库管理员(DBA)维护  NoSQL数据库虽然没有DBMS复杂，也难以维护 |

5.试述NoSQL数据库的四大类型

答：键值数据库、列族数据库、文档数据库和图数据库

6.试述键值数据库、列族数据库、文档数据库和图数据库的适用场合和优缺点。

|  |  |  |  |
| --- | --- | --- | --- |
| 数据库 | 适用场合 | 优点 | 缺点 |
| 键值数据库 | 通过键而是通过值来查的业务 | 扩展性好，灵活性好，大量写操作时性能高 | 无法存储结构化信息，条件查询效率较低 |
| 列族数据库 | 不需要ACID事务支持的情形 | 查找速度快，可扩展性强，容易进行分布式扩展，复杂性低 | 功能较少，大都不支持强事务一致性 |
| 文档数据库 | 只在相同的文档上添加事务 | 性能好（高并发），灵活性高，复杂性低，数据结构灵活  提供嵌入式文档功能，将经常查询的数据存储在同一个文档中  既可以根据键来构建索引，也可以根据内容构建索引 | 缺乏统一的查询语法 |
| 图形数据库 | 具有高度相互关联关系的数据 | 灵活性高，支持复杂的图形算法，可用于构建复杂的关系图谱 | 复杂性高，只能支持一定的数据规模 |

7.试述CAP理论的具体含义。

答：所谓的CAP指的是：

C（Consistency）：一致性，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，或者说，所有节点在同一时间具有相同的数据

A:（Availability）：可用性，是指快速获取数据，可以在确定的时间内返回操作结果，保证每个请求不管成功或者失败都有响应；

P（Tolerance of Network Partition）：分区容忍性，是指当出现网络分区的情况时（即系统中的一部分节点无法和其他节点进行通信），分离的系统也能够正常运行，也就是说，系统中任意信息的丢失或失败不会影响系统的继续运作。

8.请举例说明不同产品在设计时是如何运用CAP理论的。



9.试述数据库的ACID四性的含义

1.原子性（Atomicity)

指事务必须是原子工作单元，对于其数据修改，要么全都执行，要么全都不执行。

2.一致性（consistency）

指事务在完成时，必须使所有的数据都保持一致状态。

1. 隔离性（Isolation)

指并发事务所做的修改必须与其他并发事务所做的修改隔离。

1. 持久性（Durability)

指事务完成之后，它对于系统的影响是永久性的，该修改即使出现致命的系统故障也将一直保持。

10.试述BASE的具体含义

BASE的基本含义是基本可用（Basically Availble）、软状态（Soft-state）和最终一致性（Eventual consistency）

11.请解释软状态、无状态、硬状态的具体含义。

“软状态（soft-state）”是与“硬状态（hard-state）”相对应的一种提法。数据库保存的数据是“硬状态”时，可以保证数据一致性，即保证数据一直是正确的。“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。

12.什么是最终一致性？

最终一致性根据更新数据后各进程访问到数据的时间和方式的不同，又可以区分为：

1. 会话一致性：它把访问存储系统的进程放到会话（session）的上下文中，只要会话还存在，系统就保证“读己之所写”一致性。如果由于某些失败情形令会话终止，就要建立新的会话，而且系统保证不会延续到新的会话；
2. 单调写一致性：系统保证来自同一个进程的写操作顺序执行。系统必须保证这种程度的一致性，否则就非常难以编程了
3. 单调读一致性：如果进程已经看到过数据对象的某个值，那么任何后续访问都不会返回在那个值之前的值
4. 因果一致性：如果进程A通知进程B它已更新了一个数据项，那么进程B的后续访问将获得A写入的最新值。而与进程A无因果关系的进程C的访问，仍然遵守一般的最终一致性规则
5. “读己之所写”一致性：可以视为因果一致性的一个特例。当进程A自己执行一个更新操作之后，它自己总是可以访问到更新过的值，绝不会看到旧值
6. 试述不一致性窗口的含义。

所有后续的访问都可以读取到操作OP写入的最新值。从OP操作完成到后续访问可以最终读取到OP写入的最新值，这之间的时间间隔称为“不一致性窗口”。

14最终一致性根据更新数据后各进程访问到数据的时间和方式的不同，又可以分为哪些不同类型的一致性？

会话一致性、单调写一致性、单调写一致性、因果一致性和“读己之所写”一致性。

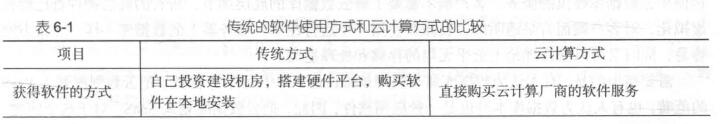
1. 什么是NewSQL数据库？

NewSQL是对各种新的可扩展、高性能数据库的简称，这类数据库不仅具有NoSQL对海量数据的存储管理能力，还保持了传统数据库支持ACID和SQL特性。

**第六章**

1.试述云数据库的概念。

答：云数据库是部署和虚拟化在云计算环境中的数据库。云数据库是在云计算的大背景下发展起来的一种新兴的共享基础架构的方法，它极大地增强了数据库的存储能力，消除了人员、硬件、软件的重复配置，让软、硬件升级变得更加容易，同时，也虚拟化了许多后端功能。云数据库具有高可扩展性、高可用性、采用多租形式和支持资源有效分发等特点。  
2.与传统的软件使用方式相比，云计算这种模式具有哪些明显的优势?





3.云数据库有哪些特性?

答：1）动态可扩展 2)高可用性 3)较低的使用代价

4)易用性 5)高性能 6)免维护 7)安全

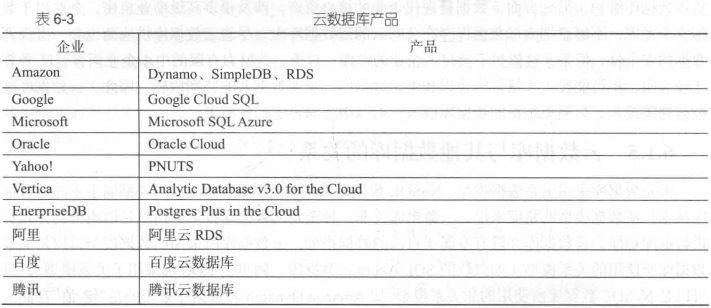
4.试述云数据库的影响。

答： 在大数据时代，每个企业几乎每天都在不断产生大量的数据。企业类型不同，对于存储的需求也千差万别，而云数据库可以很好地满足不同企业的个性化存储需求。  
 首先，云数据库可以满足大企业的海量数据存储需求。云数据库在当前数据爆炸的大数据时代具有广阔的应用前景。传统的关系数据库难以水平扩展，相本无法存储如此海量的数据。因此，具有高可扩展性的云数据库就成为企业海量数据存储管理的很好选择。  
  其次，云数据库可以满足中小企业的低成本数据存储需求。中小企业在IT 基础设施方面的投人比较有限，非常渴望从第三方方便、快捷、廉价地获得数据库服务。云数据库采用多租户方式同时为多个用户提供服务，降低了单个用户的使用成本，而且用户使用云数据库服务通常按需付费，不会浪费资源造成额外支出，因此，云数据库使用成本很低，对于中小企业而言可以大大降低企业的信息化门槛，让企业在付出较低成本的同时，获得优质的专业级数据库服务,从而有效提升企业信息化水平。  
 另外，云数据库可以满足企业动态变化的数据存储需求。企业在不同时期需要存储的数据量是不断变化的，有时增加，有时减少。在小规模应用的情况下，系统负载的变化可以由系统空闲的多余资源来处理,但是，在大规模应用的情况下，传统的关系数据库由于其伸缩性较差，不仅无法满足应用需求，而且会给企业带来高昂的存储成本和管理开销。而云数据库的良好伸缩性，可以让企业在需求增加时立即获得数据库能力的提升,在需求减少时立即释放多余的数据库能力,较好地满足企业的动态数据存储需求。

5.举例说明云数据库厂商及其代表性产品。

答：云数据库供应商主要分为三类。

1. 传统的数据库厂商，如Teradata、Oracle、IBM DB2和Microsoft SQL Server等。
2. 涉足数据库市场的云供应商，如Amazon、Google.Yahoo!、阿里、百度、腾讯等。
3. 新兴厂商，如IVertica.LongJump 和EnterpriseDB等。



7.试述UMP 系统的功能。

答：   
 UMP系统是构建在一个大的集群之上的，通过多个组件的协同作业，整个系统实现了对用户透明的 容灾、读写分离、分库分表、资源管理、资源调度、资源隔离和数据安全功能。  
  1.容灾  
  云数据库必须向用户提供一直可用的数据库连接，当MySQL实例发生故障时，系统必须自动执行故障恢复，所有故障处理过程对于用户而言是透明的，用户不会感知到后台发生的一切。  
  为了实现容灾，UMP系统会为每个用户创建两个MySQL实例，一个是主库，一个是从库，而且，这两个MySQL 实例之间互相把对方设置为备份机，任意一个MySQL实例上面发生的更新都会复制到对方。同时，Proxy服务器可以保证只向主库写人数据。  
  
  2.读写分离  
  由于每个用户都有两个MySQL实例，即主库和从库，因此，可以充分利用主从库实现用户读写操作的分离，实现负载均衡。UMP系统实现了对于用户透明的读写分离功能，当整个功能被开启时，负责向用户提供访问MySQL数据库服务的Proxy 服务器，就会对用户发起的SQL 语句进行解析，如果属于写操作,就直接发送到主库，如果是读操作，就会被均衡地发送到主库和从库上执行。

  3.分库分表  
  UMP支持对用户透明的分库分表(Shard/Horizontal Partition)。但是，用户在创建账号的时候需要指定类型为多实例，并且设置实例的个数，系统会根据用户设置来创建多组MySQL实例。除此以外，用户还需要自己设定分库分表规则，如需要确定分区字段，也就是根据哪个字段进行分库分表，还要确定分区字段里的值如何映射到不同的MySQL 实例上。  
  4.资源管理  
  UMP系统采用资源池机制来管理数据库服务器上的CPU、内存、磁盘等计算资源，所有的计算资源都放在资源池内进行统一分配，资源池是为MySQL 实例分配资源的基本单位。整个集群中的所有服务器会根据其机型、所在机房等因素被划分为多个资源池，每台服务器会被加人到相应的资源池。在资源池划分的基础上，UMP还在每台服务器内部采用Cgroup将资源进一步地细化，从而可以限制每个进程组使用资源的上限，同时保证进程组之间相互隔离。  
  5.资源调度  
  UMP系统中有3种规格的用户，分别是数据量和流量比较小的用户、中等规模用户以及需要分库分表的用户。多个小规模用户可以共享同一个MySQL实例。对于中等规模的用户，每个用户独占个MySQL 实例。用户可以根据自己的需求来调整内存空间和磁盘空间，如果用户需要更多的资源，就可以迁移到资源有空闲或者具有更高配置的服务器上对于分库分表的用户，会占有多个独立的MySQL 实例，这些实例既可以共存在同一台物理机上，也可以每个实例独占一台物理机。  
  UMP通过MySQL实例的迁移来实现资源调度。借助于阿里集团中间件团队开发的愚公系统，UMP 可以实现在不停机的情况下动态扩容、缩容和迁移。  
  6.资源隔离  
  当多个用户共享同一个MySQL 实例或者多个MySQL 实例共存在同一个物理机上时，为了保护用户应用和数据的安全，必须实现资源隔离，否则，某个用户过多消耗系统资源会严重影响到其他用户的操作性能。

7.数据安全  
 数据安全是让用户放心使用云数据库产品的关键，尤其是企业用户，数据库中存放了很多业务数据，有些属于商业机密，一旦泄露，会给企业造成损失。UMP 系统设计了多种机制来保证数据安全。

1. SSL 数据库连接。
2. 数据访问IP 白名单。
3. 记录用户操作日志。
4. SQL拦截。

13.  UMP 系统是如何保障数据安全的?

答：

1. SSL 数据库连接。  SSL (Secure Sockets Layer) 是为网络通信提供安全及数据完整性的一种安全协议，它在传输层对网络连接进行加密。Proxy 服务器实现了完整的MySQL 客户端服务器协议，可以与客户端之间建立SSL 数据库连接。
2. 数据访问IP 白名单。可以把允许访问云数据库的IP 地址放入“白名单”，只有白名单内的IP地址才能访问，其他IP地址的访问都会被拒绝，从而进一步保证账户安全。
3. 记录用户操作日志。  用户的所有操作记录都会被记录到日志分析服务器，通过检查用户操作记录，可以发现隐藏的安全漏洞。
4. SQL拦截。Proxy 服务器可以根据要求拦截多种类型的SQL 语句，比如全表扫描语句"select \*”。

15.  简述RDS 中实例与数据库的概念。

答：  RDS实例或简称“实例”，是用户购买RDS服务的基本单位。在实例中可以创建多个数据库，可以使用常见的数据库客户端连接、管理及使用数据库。可以通过RDS管理控制台或OPEN API来创建、修改和删除数据库。各实例之间相互独立、资源隔离，相互之间不存在CPU、内存、IOPS等抢占问题。但是，同一实例中的不同数据库之间是资源共享的。每个实例拥有其自己的特性，如数据库类型、版本等，系统有相应的参数来控制实例行为。用户所购买RDS实例的性能，取决于购买RDS实例时所选择的配置，可供用户选择的硬件配置项为内存和磁盘容量。

RDS数据库或简称“数据库”，是用户在一个实例下创建的逻辑单元，一个实例可以创建多个数据库，在实例内数据库命名唯一，所有数据库都会共享该实例下的资源，如CPU、内存、磁盘容量等。RDS不支持使用标准的SQL 语句或客户端工具创建数据库，必须使用OPEN API或RDS管理控制台进行操作。  
  
16.  列举连接RDS for MySQL 数据库的4 种方法。

答：  方法1: 使用客户端MySQL-Front访问。使用客户端MySQL-Front,在连接Host 框中输人数据实例链接地址、端口(默认3306)、数据库用户名和数据库密码后，单击“确定”按钮即可。  
  方法2: 使用数据库管理T 具Navicat MySQL。Navicat\_MySQL 是一套专为MySQL 设计的强大的数据库管理及开发工具，可以在连接输人框中输人数据实例地址、端口(默认3306 )、数据库用户名和数据库密码后，单击“确定”按钮即可。  
  方法3: 使用MySQL 命令登录。用户安装MySQL 客户端后，可进人命令行方式连接数据库。命令格式如下。  
  mysql -u user\_name -h yuqianli.mysql.rds.aliyuncs.com -P3306 -pxxxx

其中，-u指定的是用户名，-h指定的是主机名，-P指定的是端口，-p指定的是密码。  
  方法4: 使用阿里云控制台iDB Cloud访问。阿里云控制台iDB Cloud的页面如图6-7 所示，RDS 连接地址以及端口不需要再输人，只需在“用户名”中输人数据库的账号,在“密码”栏中输人数据库账号的密码，便可以登录RDS进行数据操作了。

**第七章**

1.试述MapReduce和Hadoop的关系。

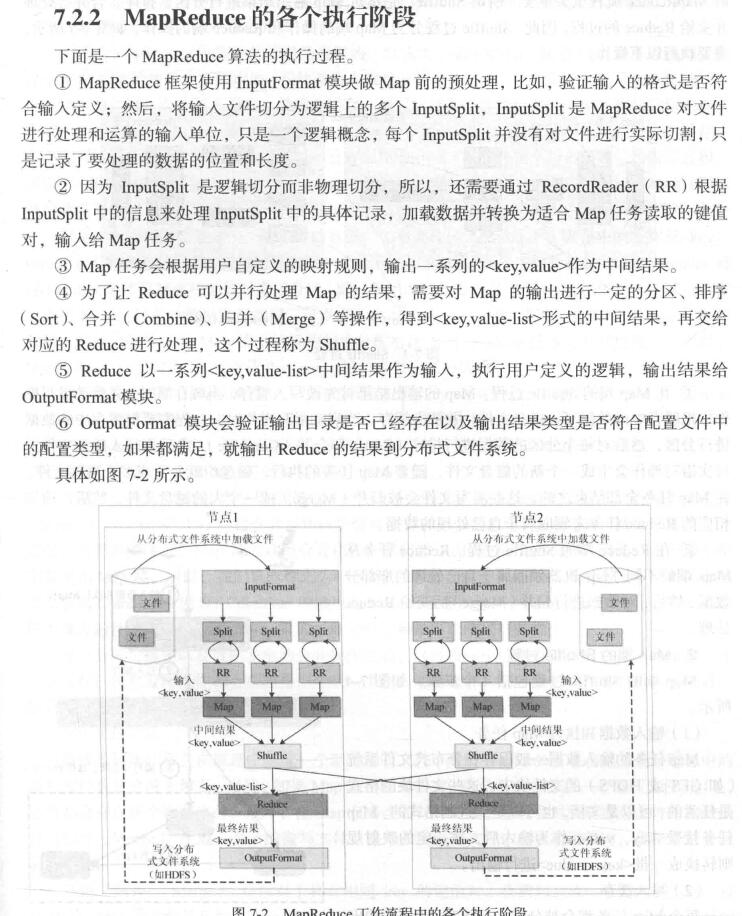
答：  谷歌公司最先提出了分布式并行编程模型MapReduce, Hadoop MapReduce是它的开源实现。谷歌的MapReduce运行在分布式文件系统GFS上，与谷歌类似，HadoopMapReduce运行在分布式文件系统HDFS上。相对而言，HadoopMapReduce 要比谷歌MapReduce 的使用门槛低很多，程序员即使没有任何分布式程序开发经验，也可以很轻松地开发出分布式程序并部署到计算机集群中。

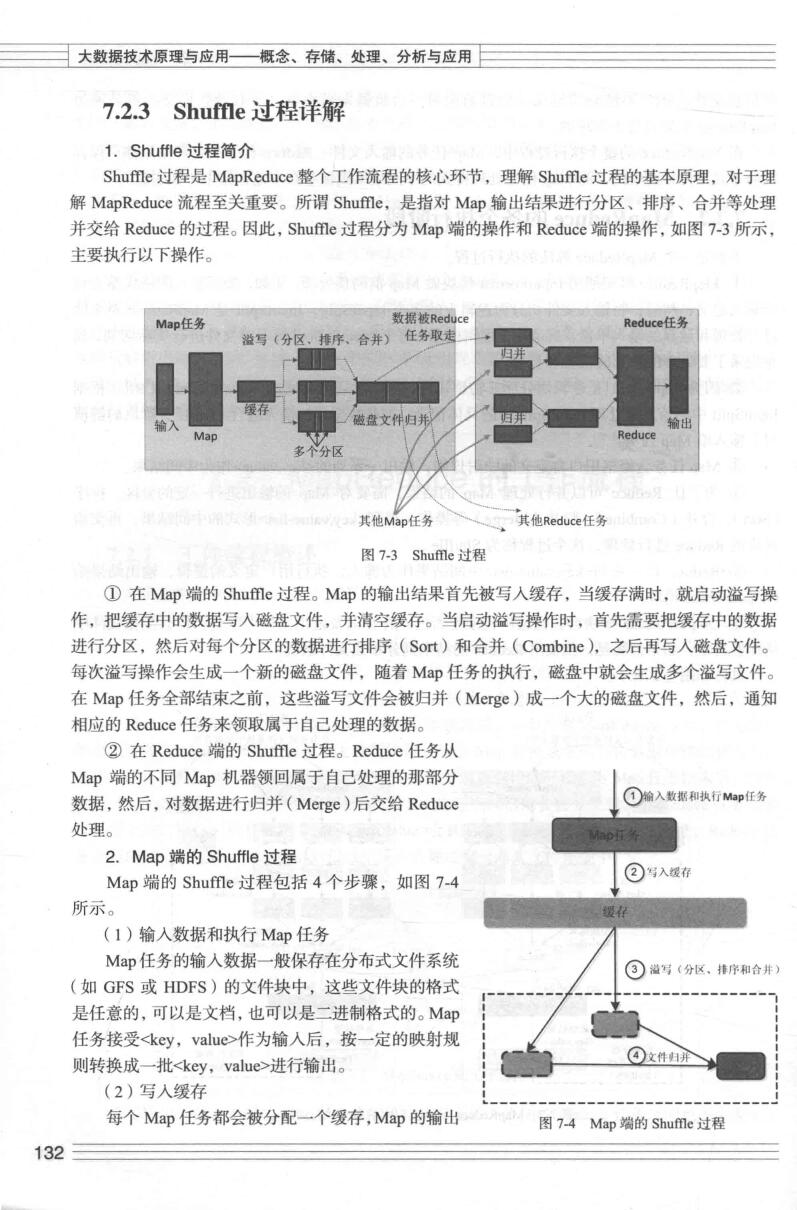
2.MapReduce 是处理大数据的有力工具，但不是每个任务都可以使用MapReduce 来进行处理。试述适合用MapReduce来处理的任务或者数据集需满足怎样的要求。

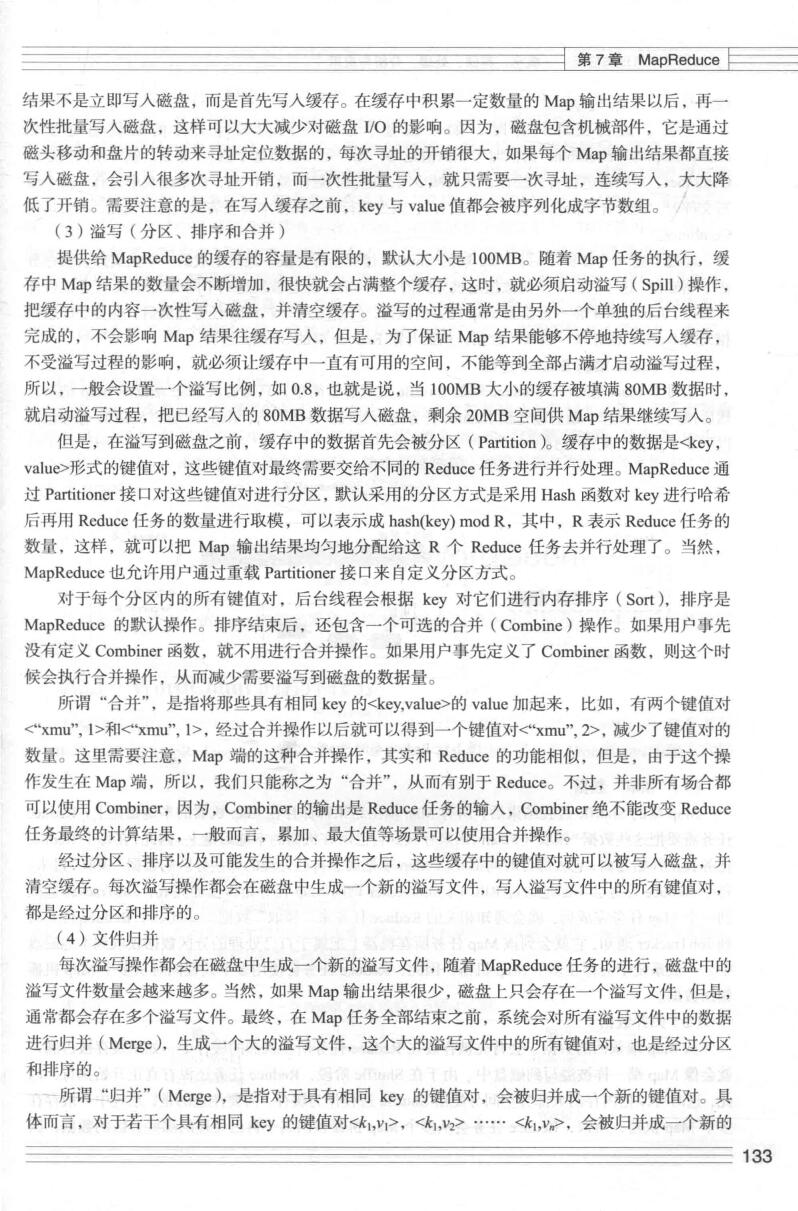
答： 适合用MapReduce来处理的数据集，需要满足一个前提条件: 待处理的数据集可以分解成许多小的数据集，而且每一个小数据集都可以完全并行地进行处理。  
  
3. MapReduce模型采用Master(JobTracker)-Slave(TaskTracker)结构，试描述JobTracker和TasKTracker的功能。

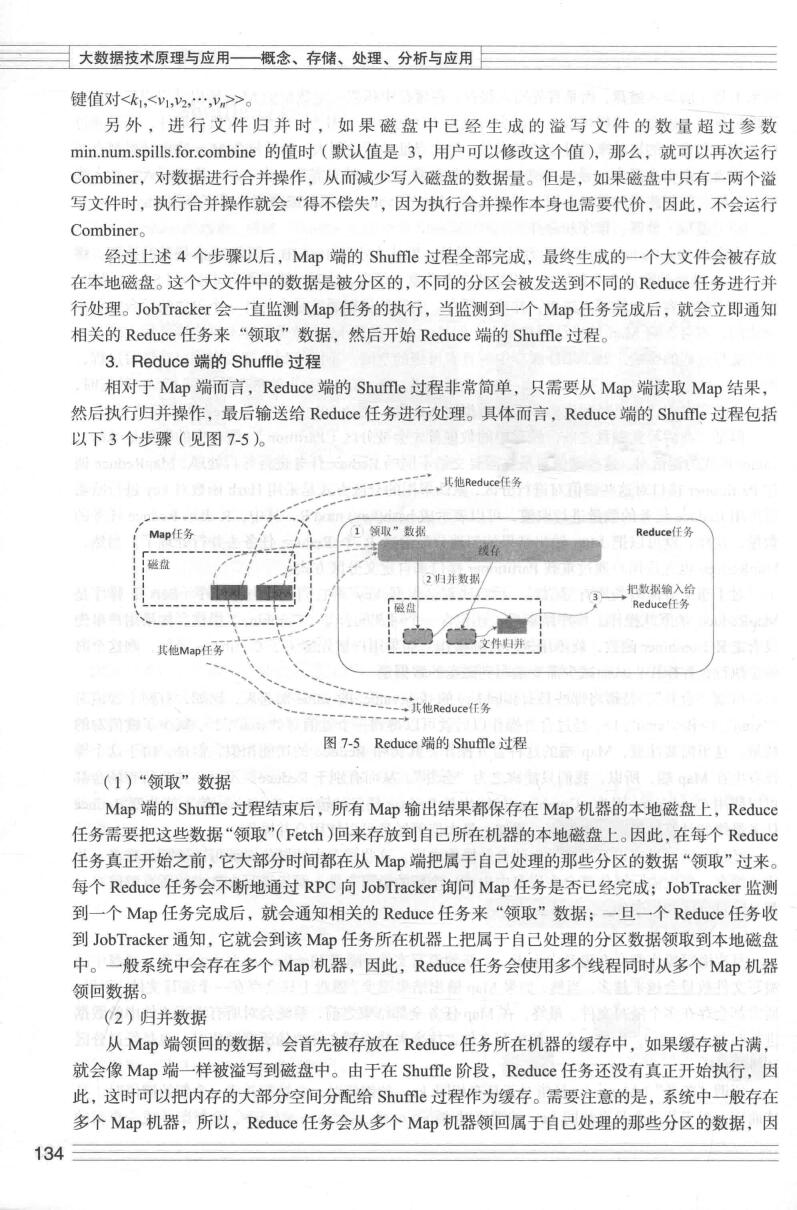
答：  MapReduce 框架采用了Master/Slave 架构，包括一个Master 和若干个Slave。Master 上运行JobTracker,Slave 上运行TaskTrackero 用户提交的每个计算作业，会被划分成若千个任务。JobTracker 负责作业和任务的调度，监控它们的执行，并重新调度已经失败的任务。TaskTracker负责执行由JobTracker指派的任务。

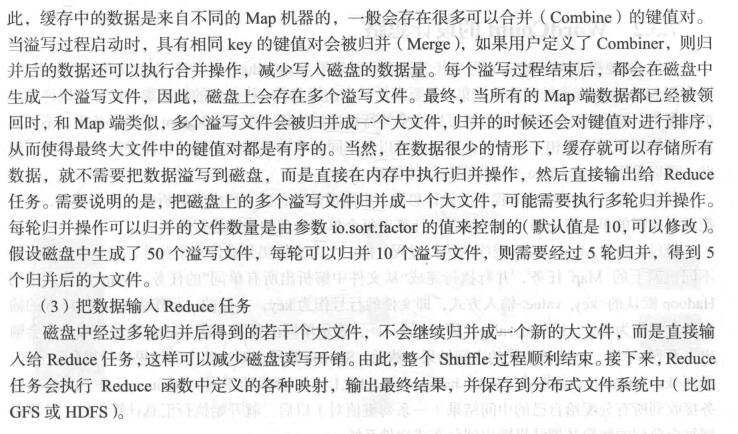
6.试述MapReduce的工作流程(需包括提交任务、Map、Shuffle、Reduce的过程)。











9.MapReduce中有这样一个原则:移动计算比移动数据更经济。试述什么是本地计算，并分析为何要采用本地计算。

答：  MapReduce设计的一个理念就是“计算向数据靠拢”，而不是“数据向计算靠拢”，因为移动数据需要大量的网络传输开销，尤其是在大规模数据环境下，这种开销尤为惊人，所以，移动计算要比移动数据更加经济。

本地计算：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，即将计算节点和存储节点放在一起运行，从而减少了节点间的数据移动开销。

10.试说明一个MapReduce程序在运行期间，所启动的Map任务数量和Reduce 任务数量各是由什么因素决定的。

11.是否所有的MapReduce程序都需要经过Map和Reduce这两个过程?如果不是，请举例说明。

答：不是。对于关系的选择运算，只需要Map过程就能实现，对于关系R 中的每个元组t,检测是否是满足条件的所需元组，如果满足条件，则输出键值对<,>,也就是说，键和值都是t。这时的Reduce函数就只是一个恒等式，对输入不做任何变换就直接输出。

12.试分析为何采用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：  对于每个分区内的所有键值对，后台线程会根据key 对它们进行内存排序(Sort ),排序是MapReduce 的默认操作。排序结束后，还包含一个可选的合并(Combine )操作。如果用户事先没有定义Combiner 函数，就不用进行合并操作。如果用户事先定义了Combiner 函数，则这个时候会执行合并操作，从而减少需要溢写到磁盘的数据量。  
  所谓“合并”，是指将那些具有相同key 的<key,value>的value 加起来，比如，有两个键值对<\*xmu",1>和<\*xmu",1>,经过合并操作以后就可以得到一个键值对<\*xmu",2>,减少了键值对的数量。

不过，并非所有场合都可以使用Combiner,因为，Combiner的输出是Reduce任务的输人，Combiner绝不能改变Reduce任务最终的计算结果，一般而言，累加、最大值等场景可以使用合并操作。

13.MapReduce程序的输入文件、输出文件都存储在HDFS中，而在Map任务完成时的中间结果则存储在本地磁盘中。试分析中间结果存储在本地磁盘而不是HDFS上有何优缺点。  
答：