大数据面笔试 题库

日期：2022.04.22

《大数据面笔试题库》

提示：

1、 题目类型包括：单选题、多选题、简答题、填空题、编程题等

2、 题目前缀【类型】【知识点】

知识点目录			
日志收集系统 Flume	HDFS	消息队列 Kafka	ETL 开源工具 Kettle
数据迁移工具 Sqoop	HBase	Redis	MongoDB
实时计算框架	Zookeeper	Hive	MapReduce
Spark	Neo4j	Hama	数据挖掘基础
数据挖掘工具	关联规则	回归算法	分类方法
Web 挖掘技术	可视化基础	D3.js	Echarts.js
企业级大数据平	Tableau	……	……

- 业务场景

1.基因生物大数据

面试官：你写的项目中，哪个项目你觉得你成长最大，介绍一下你做的这个项目。

应聘者：基因医疗大数据系统是针对基因疾病持续增长态势下面向医生、患者、相关科研人员提供人类基因数据分析、肿瘤分析、罕见病分析、病毒变异分析、基因疾病预测、疾病咨询科普的全方位、一体化的分析预测大数据平台。我们搜集了 HGNC 人类基因组数据库、COSMIC 最大的癌症基因突变数据库、DisGeNet 疾病相关的基因与突变位点数据库、Orphanet 罕见病数据库、PharmGKB 遗传药理学和药物基因组学数据库的基因疾病相关数据，使用 Spark、SparkSQL、机器学习算法从基因、罕见病、癌症、病毒等多个类别的二十多个维度进行数据挖掘、计算、分析和建模，用 Echarts 将分析结果进行可视化，将结果更清晰直观的展现出来，并实现了基因疾病的在线实时预测模块和咨询科普模块，在线实时预测模块给予用户早期发现可能患有的疾病进行指导，咨询科普模块给予用户智能咨询和疾病相关的知识图谱数据，给用户提供疾病治疗等相关的知识。

面试官：你在这个项目中主要运用了什么技术？

应聘者：这个项目采用了前后端分离的架构，有效的进行解耦，开发效率提高，为以后的大型分布式架构、弹性计算架构、微服务架构、多端化服务打下坚实的基础，后端运用了 SpringBoot、Flask，前端运用了 Vue.js 和

Node.js 以及 Ajax 异步加载技术、Echarts 数据可视化图表库、ElementUI 组件库、Ant Design 组件库。实时计算系统运用了 Flume 数据采集工具采集变更数据、Kafka 作为消息中间件、Flink 进行实时计算，使用决策树机器学习算法实现了基因疾病预测功能。

面试官：以某个模块为例，简单描述一下这个模块开发的一个流程。

应聘者：以基因组数据分析为例：

数据的获取：通过 Scrapy 爬虫，网上相关数据库的下载以及联系专业机构获取。数据处理：通过 Spark 筛除不需要的字段，Sparksql 获取对应可视化模块的数据，去除空值、排序等，将处理好的数据存入数据库。降低耦合：选择前后端分离，高效的开发项目。形成 url 接口：后端运用 Springboot 框架，获取数据库中的数据并形成独立的接口。可视化：前端采用 Vue 框架，将模块分成组件管理。通过接口获取到数据，将数据通过 ElementuUI，Ant Design 组件以及 echarts 图表展示出来。

面试官：在这个项目中，你们使用 Flume 采集数据，那么如何解决 Flume 的丢包问题。

应聘者：我们在 Flume 工作过程中，会对业务日志进行监控，例如 Flume agent 中有多少条日志，Flume 到 Kafka 后有多少条日志等等，如果数据丢失保持在 1%左右是没有问题的，当数据丢失达到 5%左右时就必须采取相应措施。

面试官：请简单描述一下如何保障 kafka 的数据不丢失。

应聘者：一个是生产者端，一个消费者端，一个 broker 端。

生产者数据的不丢失:

kafka 的 ack 机制:在 kafka 发送数据的时候，每次发送消息都会有一个确认反馈机制，确保消息正常的能够被收到，其中状态有 0，1，-1。

如果是同步模式:

ack 设置为 0，风险很大，一般不建议设置为 0。即使设置为 1，也会随着 leader 宕机丢失数据。所以如果要严格保证生产端数据不丢失，可设置为-1。

如果是异步模式:

也会考虑 ack 的状态，除此之外，异步模式下的有个 buffer，通过 buffer 来进行控制数据的发送，有两个值来进行控制，时间阈值与消息的数量阈值，如果 buffer 满了数据还没有发送出去，有个选项是配置是否立即清空 buffer。可以设置为-1，永久阻塞，也就数据不再生产。异步模式下，即使设置为-1。也可能因为程序员的不科学操作，操作数据丢失，比如 kill-9，但这是特别的例外情况。

消费者数据的不丢失:

通过 offset commit 来保证数据的不丢失，kafka 自己记录了每次消费的 offset 数值，下次继续消费的时候，会接着上次的 offset 进行消费。

而 offset 的信息在 kafka0.8 版本之前保存在 zookeeper 中，在 0.8 版本之后保存到 topic 中，即使消费者在运行过程中挂掉了，再次启动的时候会找到 offset 的值，找到之前消费消息的位置，接着消费，由于 offset 的信息写入的时候并不是每条消息消费完成后都写入的，所以这种情况有可能会造成重复消费，但是不会丢失消息。

唯一例外的情况是，我们在程序中给原本做不同功能的两个 consumer 组设置 KafkaSpoutConfig.bulider.setGroupid 的时候设置成了一样的 groupid，这种情况会导致这两个组共享同一份数据，就会产生组 A 消费 partition1，partition2 中的消息，组 B 消费 partition3 的消息，这样每个组消费的消息都会丢失，都是不完整的。为了保证每个组都独享一份消息数据，group id 一定不要重复才行。

kafka 集群中的 **broker** 的数据不丢失:

每个 broker 中的 partition 我们一般都会设置有 replication(副本)的个数，生产者写入的时候首先根据分发策略(有 partition 按 partition，有 key 按 key，都没有轮询)写入到 leader 中，follower(副本)再跟 leader 同步数据，这样有了备份，也可以保证消息数据的不丢失。

面试官：在进行实时计算式，为什么选择 Flink，Flink 相比 SparkStreaming 有什么区别?

应聘者：架构模型上:SparkStreaming 的 task 运行依赖 driver 和 executor 和 worker，当然 driver 和 excutor 还依赖于集群管理器 Standalone 或者 yarn 等。而 Flink 运行时主要是 JobManager、TaskManage 和 TaskSlot。另外一个最核心的区别是:SparkStreaming 是微批处理，运行的时候需要指定批处理的时间，每次运行 job 时处理一个批次的数据;Flink 是基于事件驱动的，事件可以理解为消息。事件驱动的应用程序是一种状态应用程序，它会从一个或者多个流中注入事件，通过触发计算更新状态，或外部动作对注入的事件作出反应。

任务调度上:SparkStreaming 的调度分为构建 DGA 图，划分 stage，生成 taskset，调度 task 等步骤，而 Flink 首先会生成 StreamGraph，接着生成 JobGraph，然后将 jobGraph 提交给 Jobmanager 由它完成 jobGraph 到 ExecutionGraph 的转变，最后由 jobManager 调度执行。

时间机制上:flink 支持三种时间机制事件时间，注入时间，处理时间，同时支持 watermark 机制处理滞后数据。SparkStreaming 只支持处理时间，Structuredstreaming 则支持了事件时间和 watermark 机制。

容错机制上:二者保证 exactly-once 的方式不同。sparkstreaming 通过保存 offset 和事务的方式;Flink 则使用两阶段提交协议来解决这个问题。

面试官：在进行预测时，为什么选择决策树算法？

应聘者：决策树是通过一系列规则对数据进行分类的过程。它提供了一种类似规则的方法是在什么条件下会得到什么样值。我们采用的是其中的分类树，是对离散变量做决策的方法。在模型训练过程中发现该方法的准确率相较于其方法更高。该方法对数据的每个特征都单独处理，在相对短的时间内能够对大型数据源做出可行且效果良好

的结果。相对于其他方法效率更高，决策树只需要构建一次，就可以反复使用，每一次预测次数都可以控制在决策树的深度内。

面试官：你在整个项目开发中遇到了什么困难，你怎么解决的。

应聘者：在项目开发过程中，利用 Echarts 做数据可视化时在 Vue 中直接在组件中引入图表的 js，结果图表不能够成功挂载，经查阅资料要等 dom 元素已经挂载到页面中，在 mounted 生命周期函数中实例化 echarts 对象。另外实时计算的 Flume 数据采集，Flume 没有原生的 MySQL source，最后我们引入 flume-ng-sql-source 插件，监控数据变化。

2.智慧金融信贷大数据

面试官：你写的项目中，哪个项目你觉得你成长最大，介绍一下你做的这个项目。

应聘者：智慧金融信贷分析管理平台是针对当前严峻的个人信用贷款情况建立的高效率、全方位、一体化监测平台。该系统主要包含首页、信贷金融分析、交易状态分析、信用额度分析、贷款预测、贷款申请六大模块。我们对大量申请信用贷款人的各项指标进行分析，采集了他们的年龄、性别、工作年限、逾期次数、贷款状态等 74 项可能影响因素，以此来了解不良信贷的原因以及最佳度量指标，通过对个人贷款信息进行分析，并将此作为模型，实现了在线实时预测模块，让银行和投资者可以随时检测出该用户的信贷风险等级。智慧金融信贷分析管理平台的开发可以帮助银行、网贷公司、金融公司、投资公司等了解申请信用贷款用户的信贷相关数据和信贷风险等级，平台产生的预测结果报告为银行和企业的贷款发放提供建议，帮助银行和企业加强风险防范。

面试官：你在这个项目中主要运用了什么技术？

应聘者：这个项目后端运用了 SpringBoot，前端运用了 Ajax 异步加载技术、Echarts 数据可视化图表库。实时计算系统运用了 Flume 数据采集工具采集变更数据、Kafka 作为消息中间件。SparkStreaming 消费 Kafka 数据进行实时计算，使用随机森林机器学习算法实现了贷款利率预测功能。

面试官：以某个模块为例，简单描述一下这个模块开发的一个流程。

应聘者：以信用额度分析模块为例：

数据的获取：通过 Scrapy 爬虫和网上相关数据库的下载；数据处理：Spark 将无关的数据筛除，Sparksql 获取对应可视化模块的数据，去除空值、排序等，将处理好的数据存入数据库；形成 url 接口：后端 Springboot 框架获取数据库中的数据并形成接口。可视化：前端通过接口获取到数据，将数据通过 Echarts 图表展示出来。

面试官：在这个项目中，使用了 flume+kafka,那么请简单描述一下 Flume 和 kafka 采集日志区别，采集日志时中间停了，怎么记录之前的日志？

应聘者：Flume 采集日志是通过流的方式直接将日志收集到存储层，而 kafka 是将缓存在 kafka 集群，待后期可以采集到存储层。

Flume 采集中间停了，可以采用文件的方式记录之前的日志，而 kafka 是采用 offset 的方式记录之前的日

志。

面试官：请简单描述一下编写 Spark Streaming 程序的基本步骤分为几步，具体有哪些？

应聘者：

1.通过创建输入 DStream 来定义输入源

2.通过对 DStream 应用转换操作和输出操作来定义流计算。

3.用 streamingContext.start()来开始接收数据和处理流程。

4.通过 streamingContext.awaitTermination()方法来等待处理结束（手动结束或因为错误而结束）。

5.可以通过 streamingContext.stop()来手动结束流计算进程。

面试官：在进行预测时，为什么选择随机森林算法？

应聘者：随机森林是通过集成学习的思想,将多棵决策树进行集成的算法。我们选择随机森林算法是因为经过多个算法对比，随机森林算法的预测准确度最高，我们训练的数据集比较大，而随机森林能够有效地在大数据集上运行，有很好的抗噪声能力。

3.基于大数据技术的机动车辆保险管理平台

面试官：首先介绍下你的项目情况。

应聘者：随着经济的发展，机动车辆的数量不断增加。当前，机动车辆保险已成为中国财产保险业务中最大的险种。因为对险种不了解，保险公司如何选择，现有的机动车险市场不熟悉等因素影响，用户不容易购买到自己心仪且确实需要的车险搭配。基于以上情况，推出了基于大数据技术的机动车辆保险管理平台。层次规划为前端可视化和后台大数据分析，应用设计方面是前端 JavaWeb 结合后端 Spark+MySQL+hdfs+Hive，前端展现相关数据信息分析结果，预测用户购买车险的概率及提出对应的车险购买方案

面试官：我看在项目中，有提到使用 hive，那么 hive 的内部表和外部表有什么区别？

应聘者：内部表：当我们删除一个管理表时，Hive 也会删除这个表中数据。管理表不适合和其他工具共享数据；外部表：删除该表并不会删除掉原始数据，删除的是表的元数据

面试官：hive 中有哪些排序方法，之间的区别说一说；

应聘者：Sort By：分区内有序；Order By：全局排序，只有一个 Reducer；Distrbute By：类似 MR 中 Partition，进行分区，结合 sort by 使用；Cluster By：当 Distribute by 和 Sorts by 字段相同时，可以使用 Cluster by 方式。Cluster by 除了具有 Distribute by 的功能外还兼具 Sort by 的功能。但是排序只能是升序排序，不能指定排序规则为 ASC 或者 DESC。

面试官：在项目中是否自定义过 UDF、UDTF 函数，以及用他们处理了什么问题，及自定义步骤

应聘者：自定义过。用 UDF 函数解析公共字段；用 UDTF 函数解析事件字段，自定义 UDF：继承 UDF，重写

evaluate 方法，自定义 UDTF：继承自 GenericUDTF，重写 3 个方法：initialize(自定义输出的列名和类型)，process（将结果返回 forward(result)），close。为什么要自定义 UDF/UDTF，因为自定义函数，可以自己埋点 Log 打印日志，出错或者数据异常，方便调试。

面试官：你介绍的时候说到 hdfs，那么简单说下 hdfs 的几个节点和作用；

应聘者：namenode：管理命名空间、配置副本数、处理客户端的请求等。Datanode：主要是读写数据。Scondarynamenode 主要是辅助 namanode，合并镜像和操作日志。

面试官：Yarn 的默认调度器、调度器分类、以及他们之间的区别；

应聘者：Hadoop 调度器重要分为三类：FIFO 、Capacity Scheduler（容量调度器）和 Fair Sceduler（公平调度器）；FIFO 调度器：先进先出，同一时间队列中只有一个任务在执行；容量调度器：多队列；每个队列内部先进先出，同一时间队列中只有一个任务在执行。队列的并行度为队列的个数；公平调度器：多队列；每个队列内部按照缺额大小分配资源启动任务，同一时间队列中有多个任务执行。队列的并行度大于等于队列的个数；一定要强调生产环境中不是使用的 FifoScheduler，面试的时侯会发现候选人大概了解这几种调度器的区别，但是问在生产环境用哪种，却说使用的 FifoScheduler（企业生产环境一定不会用这个调度的）；

面试官：Hive 有哪些方式保存元数据，各有哪些特点；

应聘者：Hive 支持三种不同的元存储服务器，分别为：内嵌式元存储服务器、本地元存储服务器、远程元存储服务器，每种存储方式使用不同的配置参数；（1）内嵌式元存储主要用于单元测试，在该模式下每次只有一个进程可以连接到元存储，Derby 是内嵌式元存储的默认数据库；（2）在本地模式下，每个 Hive 客户端都会打开到数据存储的连接并在该连接上请求 SQL 查询；（3）在远程模式下，所有的 Hive 客户端都将打开一个到元数据服务器的连接，该服务器依次查询元数据，元数据服务器和客户端之间使用 Thrift 协议通信；

面试官：Hive 底层与数据库交互原理；

应聘者：由于 Hive 的元数据可能要面临不断地更新、修改和读取操作，所以它显然不适合使用 Hadoop 文件系统进行存储。目前 Hive 将元数据存储在 RDBMS 中，比如存储在 MySQL、Derby 中。元数据信息包括：存在的表、表的列、权限和更多的其他信息。

面试官：Hive 中的压缩格式 TextFile、SequenceFile、RCfile 、ORCfile 各有什么区别？

应聘者：TextFile：默认格式，存储方式为行存储，数据不做压缩，磁盘开销大，数据解析开销大。 可结合 Gzip、Bzip2 使用(系统自动检查，执行查询时自动解压)，但使用这种方式，压缩后的文件不支持 split，Hive 不会对数据进行切分，从而无法对数据进行并行操作。并且在反序列化过程中，必须逐个字符判断是不是分隔符和行结束符，因此反序列化开销会比 SequenceFile 高几十倍；SequenceFile：SequenceFile 是 Hadoop API 提供的一种二进制文件支持，，存储方式为行存储，其具有使用方便、可分割、可压缩的特点；RCFile：存储方式：数据按行分块，每块按列存储。结合了行存储和列存储的优点，首先，RCFile 保证同一行的数据位于同一节点，因此元组重构的开销很低，其次，像列存储一样，RCFile 能够利用列维度的数据压缩，并且能跳过不必要的列读取；ORCFile:存储

方式：数据按行分块 每块按照列存储,压缩快 快速列存取,效率比 rcfile 高,是 rcfile 的改良版本；

4.基于 Hadoop 的大数据岗位分析

面试官：说下你的项目情况；

应聘者：针对城市、职位、学历、工作经验、公司规模、公司融资、工作技能、福利、岗位需求、薪资之间进行分析，然后精准的推荐给有需要的人；项目中主要使用 flume+kafka+spark +hive+sqoop+mysql+决策树构建模型进行数据采集分析，最终通过 Javaweb 框架+ Echarts 动态可视化界面展示；

面试官：Sqoop 底层运行的任务是什么；

应聘者：只有 Map 阶段，没有 Reduce 阶段的任务；

面试官：Flume 采集数据会不会丢失；

应聘者：不会；Channel 存储可以存储在 File 中，数据传输自身有事务

面试官：flume 管道内存，flume 宕机了数据丢失怎么解决；

应聘者：（1）Flume 的 channel 分为很多种，可以将数据写入到文件；（2）防止非首个 agent 宕机的方法数可以做集群或者主备。

面试官：flume 有哪些组件，具体是做什么的；

应聘者：（1）source：用于采集数据，Source 是产生数据流的地方，同时 Source 会将产生的数据流传输到 Channel，这个有点类似于 Java IO 部分的 Channel；（2）用于桥接 Sources 和 Sinks，类似于一个队列；（3）从 Channel 收集数据，将数据写到目标源(可以是下一个 Source，也可以是 HDFS 或者 HBase)；

面试官：flume 和 kafka 采集日志区别，采集日志时中间停了，怎么记录之前的日志；

应聘者：Flume 采集日志是通过流的方式直接将日志收集到存储层，而 kafka 是将缓存在 kafka 集群，待后期可以采集到存储层；Flume 采集中间停了，可以采用文件的方式记录之前的日志，而 kafka 是采用 offset 的方式记录之前的日志；

面试官：Spark 常用算子 reduceByKey 与 groupByKey 的区别，哪一种更具优势；

应聘者：（1）reduceByKey:按照 key 进行聚合，在 shuffle 之前有 combine(预聚合)操作，返回结果是 RDD[k,v]；（2）groupByKey:按照 key 进行分组，直接进行 shuffle。

开发指导：reduceByKey 比 groupByKey 建议使用。但是需要注意是否会影响业务逻辑。

面试官：Hive 的函数：UDF、UDAF、UDTF 的区别；

应聘者：（1）UDF：单行进入，单行输出；（2）UDAF：多行进入，单行输出；（3）UDTF：单行输入，多行输

出。

- 面试/笔试题

1.【简答题】【Kafka】为什么要使用 kafka？

缓冲和削峰：上游数据时有突发流量，下游可能扛不住，或者下游没有足够多的机器来保证冗余，kafka 在中间可以起到一个缓冲的作用，把消息暂存在 kafka 中，下游服务就可以按照自己的节奏进行慢慢处理。解耦和扩展性：项目开始的时候，并不能确定具体需求。消息队列可以作为一个接口层，解耦重要的业务流程。只需要遵守约定，针对数据编程即可获取扩展能力。冗余：可以采用一对多的方式，一个生产者发布消息，可以被多个订阅 topic 的服务消费到，供多个毫无关联的业务使用。健壮性：消息队列可以堆积请求，所以消费端业务即使短时间死掉，也不会影响主要业务的正常进行。异步通信：很多时候，用户不想也不需要立即处理消息。消息队列提供了异步处理机制，允许用户把一个消息放入队列，但并不立即处理它。想向队列中放入多少消息就放多少，然后在需要的时候再去处理它们。

2.【简答题】【Kafka】 Kafka 消费过的消息如何再消费？

kafka 消费消息的 offset 是定义在 zookeeper 中的， 如果想重复消费 kafka 的消息，可以在 redis 中自己记录 offset 的 checkpoint 点（n 个），当想重复消费消息时，通过读取 redis 中的 checkpoint 点进行 zookeeper 的 offset 重设，这样就可以达到重复消费消息的目的了。

3.【简答题】【Kafka】kafka 的数据是放在磁盘上还是内存上，为什么速度会快？

kafka 使用的是磁盘存储。

速度快是因为：

顺序写入：因为硬盘是机械结构，每次读写都会寻址->写入，其中寻址是一个“机械动作”，它是耗时的。所以硬盘 “讨厌”随机 I/O， 喜欢顺序 I/O。为了提高读写硬盘的速度，Kafka 就是使用顺序 I/O。Memory Mapped Files（内存映射文件）：64 位操作系统中一般可以表示 20G 的数据文件，它的工作原理是直接利用操作系统的 Page 来实现文件到物理内存的直接映射。完成映射之后你对物理内存的操作会被同步到硬盘上。Kafka 高效文件存储设计： Kafka 把 topic 中一个 parition 大文件分成多个小文件段，通过多个小文件段，就容易定期清除或删除已经消费完文件，减少磁盘占用。通过索引信息可以快速定位 message 和确定 response 的 大 小。通过 index 元数据全部映射到 memory（内存映射文件），可以避免 segment file 的 IO 磁盘操作。通过索引文件稀疏存储，可以大幅降低 index 文件元数据占用空间大小。

4.【简答题】【Kafka】 kafka 重启是否会导致数据丢失？

kafka 是将数据写到磁盘的，一般数据不会丢失。但是在重启 kafka 过程中，如果有消费者消费消息，那么

kafka 如果来不及提交 offset，可能会造成数据的不准确（丢失或者重复消费）。

5.【简答题】【Kafka】 kafka 数据分区和消费者的关系？

每个分区只能由同一个消费组内的一个消费者(consumer)来消费，可以由不同的消费组的消费者来消费，同组的消费者则起到并发的效果。

6.【简答题】【Kafka】Kafka 单条日志传输大小

kafka 对于消息体的大小默认为单条最大值是 1M 但是在我们应用场景中, 常常会出现一条消息大于 1M，如果不对 kafka 进行配置。则会出现生产者无法将消息推送到 kafka 或消费者无法去消费 kafka 里面的数据, 这时我们就要对 kafka 进行以下配置：

server.properties

1、replicA、fetch.max.bytes: 1048576 broker 可复制的消息的最大字节数, 默认为 1M

2、message.max.bytes: 1000012 kafka 会接收单个消息 size 的最大限制， 默认为 1M 左右

注意：message.max.bytes 必须小于等于 replicA、fetch.max.bytes，否则就会导致 replica 之间数据同步失败。

7.【简答题】【Kafka】 kafka 宕机了如何解决？

1）先考虑业务是否受到影响

kafka 宕机了，首先我们考虑的问题应该是所提供的服务是否因为宕机的机器而受到影响，如果服务提供没问题，如果实现做好了集群的容灾机制，那么这块就不用担心了。

2）节点排错与恢复

想要恢复集群的节点，主要的步骤就是通过日志分析来查看节点宕机的原因，从而解决，重新恢复节点。

8.【简答题】【Kafka】采集数据为什么选择 Flume 与 kafka?

采集层主要可以使用 Flume,Kafka 等技术。

Flume:Flume 是管道流方式，提供了很多的默认实现，让用户通过参数部署，及扩展 API。

Kafka:Kafka 是一个可持久化的分布式的消息队列。Kafka 是一个非常通用的系统。你可以有许多生产者和很多的消费者共享多个主题 Topics。相比之下,Flume 是一个专用工具被设计为旨在往 HDFS，HBase 发送数据。它对 HDFS 有特殊的优化，并且集成了 Hadoop 的安全特性。所以，Cloudera 建议如果数据被多个系统消费的话，使用 kafka;如果数据被设计给 Hadoop 使用，使用 Flume。

9.【简答题】【Kafka】kafka 数据分区和消费者的关系?

每个分区只能由同一个消费组内的一个消费者(consumer)来消费，可以由不同的消费组的消费者来消费，同组

的消费者则起到并发的效果。

10.【简答题】【Kafka】kafka 重启是否会导致数据丢失?

1.kafka 是将数据写到磁盘的，一般数据不会丢失。

2.但是在重启 kafka 过程中，如果有消费者消费消息，那么 kafka 如果来不及提交 offset，可能会造成数据的不准确(丢失或者重复消费)。

11.【简答题】【Flume】在使用 flume 采集数据的过程中，为防止数据进入管道内存时，flume 突然宕机导致数据丢失，可采用哪些策略避免？

1）Flume 的 channel 分为很多种，可以将数据写入到文件。

2）防止非首个 agent 宕机的方法数可以做集群或者主备。

12.【简答题】【Flume】数据采集时，如何解决 Flume 丢包问题？

单机 upd 的 flume source 的配置，100+M/s 数据量，10w qps flume 就开始大量丢包，因此很多公司在搭建系统时，抛弃了 Flume，自己研发传输系统，但是往往会参考 Flume 的 Source-Channel-Sink 模式；

一些公司在 Flume 工作过程中，会对业务日志进行监控，例如 Flume agent 中有多少条日志，Flume 到 Kafka 后有多少条日志等等，如果数据丢失保持在 1%左右是没有问题的，当数据丢失达到 5%左右时就必须采取相应措施。

13.【简答题】【Flume】Flume 使用场景

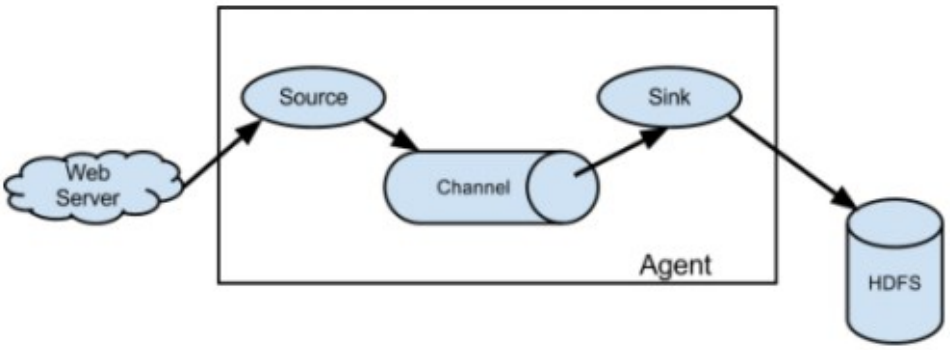
线上数据一般主要是落地（存储到磁盘）或者通过 socket 传输给另外一个系统，这种情况下，你很难推动线上应用或服务去修改接口，实现直接向 kafka 里写数据，这时候你可能就需要 flume 这样的系统帮你去做传输。

14.【简答题】【Flume】flume 不采集 Nginx 日志，通过 Logger4j 采集日志，优缺点是什么？

优点：Nginx 的日志格式是固定的，但是缺少 sessionid，通过 logger4j 采集的日志是带有 sessionid 的，而 session 可以通过 redis 共享，保证了集群日志中的同一 session 落到不同的 tomcat 时，sessionId 还是一样的，而且 logger4j 的方式比较稳定，不会宕机。

缺点：不够灵活，logger4j 的方式和项目结合过于紧密，而 flume 的方式比较灵活，拔插式比较好，不会影响项目性能。

15.【简答题】【Flume】flume 有哪些组件，flume 的 source、channel、sink 具体是做什么的？



1）source：用于采集数据，Source 是产生数据流的地方，同时 Source 会将产生的数据流传输到 Channel，这个有点类似于 Java IO 部分的 Channel。

2）channel：用于桥接 Sources 和 Sinks，类似于一个队列。

3）sink：从 Channel 收集数据，将数据写到目标源(可以是下一个 Source，也可以是 HDFS 或者 HBase)。

注意：要熟悉 source、channel、sink 的类型。

16.【简答题】【Flume】flume 管道内存，flume 宕机了数据丢失怎么解决？

Flume 的 channel 分为很多种，可以将数据写入到文件

防止非首个 agent 宕机的方法数可以做集群或者主备

17.【单选题】【Flume】在 flume 说法中，以下说法中错误的是(　　)。

A、Flume 是一个分布式、可靠、和高可用的海量日志采集、聚合和传输的系统。

B、支持在系统中定制各类数据发送方，用于收集数据。

C、提供对数据及进行简单处理，并写到各种数据接受方（可定制）的能力。

D、Flume 可以在 windows 和 unix 环境下运行。

答案：D。

1)Flume 提供一个分布式的，可靠的，对大数据量的日志进行高效收集、聚集、移动的服务，Flume 只能在 Unix 环境下运行。

2) Flume 基于流式架构，容错性强，也很灵活简单。

3) Flume、Kafka 用来实时进行数据收集，Spark、Storm 用来实时处理数据，impala 用来实时查询。

18.【单选题】【Flume】 在 flume 说法中，以下说法中错误的是(　　)

A、每个 Flume Agent 包含三个主要组件：Source、Channel、Sink。

B、Source 可以监听一个或者多个网络端口，用于接收数据或者可以从本地文件系统读取数据，每个 Source 必须至多连接三个 Channel。

C、多个 Source 可以安全的写入到相同的 Channel 读取，并且多个 Sink 可以从相同的 Channel 进行读取。

D、flume 框架对 hadoop 和 zookeeper 的依赖 jar 包上，并不要求 flume 启动时必须将 hadoop 和 zookeeper 服务也启动。

答案：B。

解析：Source 可以监听一个或者多个网络端口，用于接收数据或者可以从本地文件系统读取数据。每个 Source 必须至少连接一个 Channel。

19.【简答题】【Flume】请简述 Flume 配置文件编写步骤？

五步：

（1）定义 agent 和 sources、channels、sinks 的名称

（2）定义 sources

（3）定义 channels

（4）定义 sinks

（5）绑定 sources、channels、sink

20.【简答题】【Flume、Kafka】Flume 与 Kafka 的选取

采集层主要可以使用 Flume、Kafka 两种技术。

Flume：Flume 是管道流方式，提供了很多的默认实现，让用户通过参数部署，及扩展 API。

Kafka：Kafka 是一个可持久化的分布式的消息队列。

Kafka 是一个非常通用的系统。你可以有许多生产者和很多的消费者共享多个主题 Topics。相比之下，Flume 是一个专用工具被设计为旨在往 HDFS，HBase 发送数据。它对 HDFS 有特殊的优化，并且集成了 Hadoop 的安全特性。所以，Cloudera 建议如果数据被多个系统消费的话，使用 kafka；如果数据被设计给 Hadoop 使用，使用 Flume。

正如你们所知 Flume 内置很多的 source 和 sink 组件。然而，Kafka 明显有一个更小的生产消费者生态系统，并且 Kafka 的社区支持不好。希望将来这种情况会得到改善，但是目前：使用 Kafka 意味着你准备好了编写你自己的生产者和消费者代码。如果已经存在的 Flume Sources 和 Sinks 满足你的需求，并且你更喜欢不需要任何开发的系统，请使用 Flume。

Flume 可以使用拦截器实时处理数据。这些对数据屏蔽或者过量是很有用的。Kafka 需要外部的流处理系统才

能做到。

Kafka 和 Flume 都是可靠的系统，通过适当的配置能保证零数据丢失。然而，Flume 不支持副本事件。于是，如果 Flume 代理的一个节点奔溃了，即使使用了可靠的文件管道方式，你也将丢失这些事件直到你恢复这些磁盘。如果你需要一个高可靠性的管道，那么使用 Kafka 是个更好的选择。

Flume 和 Kafka 可以很好地结合起来使用。如果你的设计需要从 Kafka 到 Hadoop 的流数据，使用 Flume 代理并配置 Kafka 的 Source 读取数据也是可行的：你没有必要实现自己的消费者。你可以直接利用 Flume 与 HDFS 及 HBase 的结合的所有好处。你可以使用 Cloudera Manager 对消费者的监控，并且你甚至可以添加拦截器进行一些流处理。

21.【简答题】【Flume、Kafka】数据怎么采集到 Kafka？

使用官方提供的 flumeKafka 插件，插件的实现方式是自定义了 flume 的 sink，将数据从 channle 中取出，通过 kafka 的 producer 写入到 kafka 中，可以自定义分区等。

22.【单选题】【Hive】关于 Hive CRUD 操作不正确的是(　　)

A、show databases:查看所有的数据库

B、create databse bigdata;创建数据库

C、show table;查看数据库中所有的表

D、load data local inpath '/ student.txt' into table stu;从本地导入到 hive 的指定数据表中（导入数据）

答案：C。

解析：show tables;查看数据库中所有的表。

23.【单选题】【Hive】以下哪个操作的 reduce 数设置无效？(　　)

A、group by

B、order by

C、distribute by

D、Join

答案：B。

解析：order by 始终只有一个 reduce。

24.【单选题】【Hive】以下关于 Hive 基本操作描述正确的是(　　)

A、创建外部表使用 external 关键字，创建普通表需要指定 internal 关键字

B、创建外部表必须要指定 location 信息

C、加载数据到 Hive 时源数据必须是 HDFS 的一个路径

D、分区可以在创建表时指定也可在创建表后通过 Alter 命令添加分区

答案：B。

解析：EXTERNAL 关键字可以让用户创建一个外部表，在建表的同时指定一个指向实际数据的路径（LOCATION），Hive 创建内部表时，会将数据移动到数据仓库指向的路径；若创建外部表，仅记录数据所在的路径，不对数据的位置做任何改变。

25.【单选题】【Hive】下列关于分区表说法正确的是(　　)

A、Partition 对应于数据库的 Partition 列的密集索引

B、分区表跟外部表删除数据原理相似，即删除元数据信息 metastore 时，实际数据不会随着丢失

C、当删除一个分区时，不会删除实际数据

D、分区表和内部表在元数据的组织上是相同的，而实际数据的存储则有较大的差异

答案：A。

解析：当删除一个分区表时，会删除实际数据；分区表和内部表在元数据的组织上是不相同的，而实际数据的存储则差异不是很大。

26.【单选题】【Hive】关于 Hive 元数据描述错误的是（　　）

A、 Hive 的数据分为真实的数据和真实数据的元数据

B、 Hive 元数据一般存放在 HDFS 上

C、 Hive 元数据信息即数据的描述信息，比如存储位置、时间、大小之类

D、Hive 元数据信息存放配置可以通过修改 hive-site.xml

答案：B。

解析：Hive 元数据一般存放在 mysql 中。

27.【单选题】【Hive】关于 Hive 的数据导出操作不正确的是(　　)

A、insert overwrite local directory 'path' select q1;将数据保存到本地

B、insert into directory 'hdfs_path' select * from dept;将数据保存到 HDFS 上

C、可以使用 Linux 命令执行 HDFS,使用-e、-f，然后重定向到指定的文件

D、可以使用 Sqoop 框架将数据导出

答案：B。

解析：insert overwrite directory 'hdfs_path' select * from dept; 将数据保存到 HDFS 上

28.【单选题】【Hive】 关于 Hive 元数据描述错误的是（ ）

A、version：存储 Hive 的版本信息。可以有多条数据

B、存储 Hive 数据库的元数据表有 DBS、DATABASE_PARAMS

C、在存储 Hive 数据库的元数据表中 DBS 是主表，其他都是从表

D、存储表 Hive 表的元数据表 TBLS 是表的主体部分，包括表名称、创建时间、所属数据库 id、所在目录等信息

答案：A。

解析：version：存储 hive 的版本信息的表，有且只能有一条记录，多或者少都不行。

29.【单选题】【Hive】 关于 Hive 元数据描述错误的是？（ ）

A、Hive 中的元数据包括表的名字、表的列和分区及其属性，表的属性，表的数据所在目录等。

B、Hive 将表中的元数据信息存储在 derby、mysql、oracle 数据库

C、Hive 元数据表及分区的属性、存放位置等元数据存储路径和内表一样，分为本地和远程，可通过 hive-site.xml 文件设置

D、存储 Hive 数据库的元数据表有 DBS、DATABASE_PARAMS

答案：B。

解析：Hive 不支持 oracle。

30.【单选题】【Hive】 关于 metastore 常见配置错误的是（ ）

A、 hive.metastore.warehouse.dir 存储表格数据的目录

B、 hive.metastore.uris 使用内嵌的 metastore 服务

C、 javax.jdo.option.ConnectionDriverName 数据库驱动类

D、javax.jdo.option.ConnectionURL 所使用数据库的 URL

答案：B。

解析：hive.metastore.uris 如果不使用内嵌的 metastore 服务，需指定远端服务的 Uri。

31.【单选题】【Hive】在安装 Hive 时，需要在下面哪个文件配置元数据相关信息？（ ）

A、 hive-env.sh

B、 hive-site.xml

C、hive-exec-log4j.properties

D、hive-log4j.properties

答案：B。

解析：关于 Hive 的配置都在 hive-site.xml 文件中。

32.【单选题】【Hive】Hive 的元数据存储在 derby 和 mysql 中有什么区别 ？( )

A、没区别

B、多会话

C、支持网络环境

D、数据库的区别

答案：B。

解析：Hive 自带的数据库 derby 不能同时开启两个会话窗口。

33.【简答题】【Hive】某公司网站每日访问量达到 10 亿级别的访问量，每次访问记录一条数据，数据包含如下字段：用户 ID，访问时间（毫秒级），访问页面。要求使用 hive 求出所有在 5 分钟内访问次数达到 100 次的用户，请给出解题思路？

1.选出当天访问次数达到 100 次的用户（即当天有 100 及以上条数据的用户）：根据用户 ID 分组，count

2.在每个用户 ID 小组内（步骤 1 已进行分组）按访问时间进行升序排序

3.计算 time-lag(time,100)，若 time-lag(time,100)<=5601000(毫秒)，即为满足条件的用户，筛选出。

34.【简答题】【Hive】简述 Hive 排序优化的几种方式及作用范围？

Order By 全局排序，一个 Reduce

Sort By 每个 reduce 内部进行排序，不是全局排序

Distribute By 类似 MR 中 partition，进行分区，结合 sort by 使用

Cluster By　当 distribute 和 sort 字段相同时，使用方式

35.【简答题】【Hive】简述 Hive 的四种常见的数据导入方式。

Hive 的常见四种数据导入方式是：

（1）从本地文件系统中导入数据到 Hive 表；

load data local inpath '/opt/data/test.txt' into table user;

（2）从 HDFS 上导入数据到 Hive 表；

load data inpath '/iflytek/test.txt' into table user;

（3）从别的表中查询出相应的数据并导入到 Hive 表中；

insert into table test　partition (age='25')　id, name, tel> from user;

（4）在创建表的时候通过从别的表中查询出相应的记录并插入到所创建的表中。

　create table test2 as select id, name from user;

36.【简答题】【Hive】Hive 优化有哪些方法？

MapJoin

行列过滤

采用分桶技术

采用分区技术

合理设置 Map 数

通常情况下，作业会通过 input 的目录产生一个或者多个 map 任务，如果一个任务有很多小文件（远远小于块大小 128m），则每个小文件也会被当做一个块，用一个 map 任务来完成，而一个 map 任务启动和初始化的时间远远大于逻辑处理的时间，就会造成很大的资源浪费。而且，同时可执行的 map 数是受限的。

小文件进行合并:在 Map 执行前合并小文件，减少 Map 数：CombineHiveInputFormat 具有对小文件进行合并的功能（系统默认的格式）。HiveInputFormat 没有对小文件合并功能合理设置 Reduce 数。

37.【简答题】【Hive】Hive 内部表和外部表的区别？

创建表时：创建内部表时，会将数据移动到数据仓库指向的路径；若创建外部表，仅记录数据所在的路径，不对数据的位置做任何改变。

删除表时：在删除表的时候，内部表的元数据和数据会被一起删除， 而外部表只删除元数据，不删除数据。这样外部表相对来说更加安全些，数据组织也更加灵活，方便共享源数据。

38.【简答题】【Hive】请谈一下 Hive 的特点？

hive 是基于 Hadoop 的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供完整的 sql

查询功能，可以将 sql 语句转换为 MapReduce 任务进行运行。其优点是学习成本低，可以通过类 SQL 语句快速实现简单的 MapReduce 统计，不必开发专门的 MapReduce 应用，十分适合数据仓库的统计分析，但是 Hive 不支持实时查询。

39.【简答题】【Hive】所有的 Hive 任务都会有 MapReduce 的执行吗？

不是，从 Hive0.10.0 版本开始，对于简单的不需要聚合的类似 SELECT from LIMIT n 语句，不需要起 MapReduce job，直接通过 Fetch task 获取数据。

40.【简答题】【Hive】Hive 的函数：UDF、UDAF、UDTF 的区别？

UDF：单行进入，单行输出

UDAF：多行进入，单行输出

UDTF：单行输入，多行输出

41.【简答题】【Hive】hive 内部表和外部表的区别

未被 external 修饰的是内部表(managedtable)，被 external 修饰的为外部表(externaltable)

区别:

1)内部表数据由 Hive 自身管理，外部表数据由 HDFS 管理;

2)内部表数据存储的位置是 hive.metastore.warehouse.dir(默认:/user/hive/warehouse)，外部表数据的存储位置由自己制定(如果没有 LOCATION，Hive 将在 HDFS 上的/user/hive/warehouse 文件夹下以外部表的表名创建一个文件夹，并将属于这个表的数据存放在这里);

3)删除内部表会直接删除元数据(metadata)及存储数据;删除外部表仅仅会删除元数据，HDFS 上的文件并不会被删除;

42.【简答题】【Hive】hive 有索引吗

Hive 支持索引，但是 Hive 的索引与关系型数据库中的索引并不相同，比如，Hive 不支持主键或者外键。

Hive 索引可以建立在表中的某些列上，以提升一些操作的效率，例如减少 MapReduce 任务中需要读取的数据块的数量。

在可以预见到分区数据非常庞大的情况下，索引常常是优于分区的。

虽然 Hive 并不像事物数据库那样针对个别的行来执行查询、更新、删除等操作。它更多的用在多任务节点的场景下，快速地全表扫描大规模数据。但是在某些场景下，建立索引还是可以提高 Hive 表指定列的查询速度。(虽然效果差强人意)

索引适用的场景

适用于不更新的静态字段。以免总是重建索引数据。每次建立、更新数据后，都要重建索引以构建索引表。

Hive 索引的机制如下:

hive 在指定列上建立索引，会产生一张索引表(Hive 的一张物理表)，里面的字段包括，索引列的值、该值对应的 HDFS 文件路径、该值在文件中的偏移量;v0.8 后引入 bitmap 索引处理器，这个处理器适用于排重后，值较少的列(例如，某字段的取值只可能是几个枚举值)因为索引是用空间换时间，索引列的取值过多会导致建立 bitmap 索引表过大。但是，很少遇到 hive 用索引的。说明还是有缺陷 or 不合适的地方的。

43.【简答题】【Hive】sortby 和 orderby 的区别

Order by 会对输入做全局排序，因此只有一个 reducer(多个 reducer 无法保证全局有序)只有一个 reducer，会导致当输入规模较大时，需要较长的计算时间。

Sort by 不是全局排序，其在数据进入 reducer 前完成排序.

因此，如果用 sort by 进行排序，并且设置 mapreD、reduce.tasks>1，则 sortby 只保证每个 reducer 的输出有序，不保证全局有序。

44.【简答题】【Hive】在项目中是否自定义过 UDF、UDTF 函数，以及用他们处理了什么问题，及自定义步骤?

自定义过。

用 UDF 函数解析公共字段;用 UDTF 函数解析事件字段。

自定义 UDF:继承 UDF，重写 evaluate 方法自定义 UDTF:继承自 GenericUDTF，重写 3 个方法:initialize(自定义输出的列名和类型)，process(将结果返回 forward(result))，close

为什么要自定义 UDF/UDTF，因为自定义函数，可以自己埋点 Log 打印日志，出错或者数据异常，方便调试。

45.【简答题】【Hive】用过哪些 Hive 窗口函数？

RANK()排序相同时会重复，总数不会变 DENSE_RANK()排序相同时会重复，总数会减少 ROW_NUMBER()会根据顺序计算

1)OVER():指定分析函数工作的数据窗口大小，这个数据窗口大小可能会随着行的变而变化

2)CURRENTROW:当前行

3)nPRECEDING:往前 n 行数据

4)nFOLLOWING:往后 n 行数据

5)UNBOUNDED:起点，UNBOUNDEDPRECEDING 表示从前面的起点，UNBOUNDEDFOLLOWING 表示到后面的终点

6)LAG(col,n):往前第 n 行数据

7)LEAD(col,n):往后第 n 行数据

8)NTILE(n):把有序分区中的行分发到指定数据的组中，各个组有编号，编号从 1 开始，对于每一行，NTILE 返

回此行所属的组的编号。注意:n 必须为 int 类型。

46.【简答题】【Hive】Hive 和数据库比较

Hive 和数据库除了拥有类似的查询语言，再无类似之处。

1)数据存储位置

Hive 存储在 HDFS。数据库将数据保存在块设备或者本地文件系统中。

2)数据更新

Hive 中不建议对数据的改写。而数据库中的数据通常是需要经常进行修改的，

3)执行延迟

Hive 执行延迟较高。数据库的执行延迟较低。当然，这个是有条件的，即数据规模较小，当数据规模大到超过数据库的处理能力的时候，Hive 的并行计算显然能体现出优势。

4)数据规模

Hive 支持很大规模的数据计算;数据库可以支持的数据规模较小。

47.【简答题】【Hive】Hive4 个排序的区别

1)SortBy:分区内有序;

2)OrderBy:全局排序，只有一个 Reducer;

3)DistrbuteBy:类似 MR 中 Partition，进行分区，结合 sortby 使用。

4)ClusterBy:当 Distributeby 和 Sortsby 字段相同时，可以使用 Clusterby 方式。Clusterby 除了具有 Distributeby 的功能外还兼具 Sortby 的功能。但是排序只能是升序排序，不能指定排序规则为 ASC 或者 DESC。

48.【简答题】【Hive】写出 hive 中 split、coalesce 及 collect_list 函数的用法（可举例）？

split 将字符串转化为数组，即：split('a,b,c,d' , ',') ==> ["a","b","c","d"]

coalesce(T v1, T v2, …) 返回参数中的第一个非空值；如果所有值都为 NULL，那么返回 NULL

collect_list 列出该字段所有的值，不去重 select collect_list(id) from table

49.【简答题】【Hive】Hive 有哪些方式保存元数据，各有哪些特点？

Hive 支持三种不同的元存储服务器，分别为：内嵌式元存储服务器、本地元存储服务器、远程元存储服务器，每种存储方式使用不同的配置参数；

内嵌式元存储主要用于单元测试，在该模式下每次只有一个进程可以连接到元存储，Derby 是内嵌式元存储的默认数据库；

在本地模式下，每个 Hive 客户端都会打开到数据存储的连接并在该连接上请求 SQL 查询；

在远程模式下，所有的 Hive 客户端都将打开一个到元数据服务器的连接，该服务器依次查询元数据，元数据

服务器和客户端之间使用 Thrift 协议通信。

50.【简答题】【Hive】Hive 底层与数据库交互原理？

由于 Hive 的元数据可能要面临不断地更新、修改和读取操作，所以它显然不适合使用 Hadoop 文件系统进行存储。目前 Hive 将元数据存储在 RDBMS 中，比如存储在 MySQL、Derby 中。元数据信息包括：存在的表、表的列、权限和更多的其他信息。

51.【简答题】【Hive】Hive 中的压缩格式 TextFile、SequenceFile、RCfile 、ORCfile 各有什么区别？

TextFile：默认格式，存储方式为行存储，数据不做压缩，磁盘开销大，数据解析开销大。 可结合 Gzip、Bzip2 使用(系统自动检查，执行查询时自动解压)，但使用这种方式，压缩后的文件不支持 split，Hive 不会对数据进行切分，从而无法对数据进行并行操作。并且在反序列化过程中，必须逐个字符判断是不是分隔符和行结束符，因此反序列化开销会比 SequenceFile 高几十倍

SequenceFile：SequenceFile 是 Hadoop API 提供的一种二进制文件支持，，存储方式为行存储，其具有使用方便、可分割、可压缩的特点

RCFile：存储方式：数据按行分块，每块按列存储。结合了行存储和列存储的优点，首先，RCFile 保证同一行的数据位于同一节点，因此元组重构的开销很低，其次，像列存储一样，RCFile 能够利用列维度的数据压缩，并且能跳过不必要的列读取，RCFile 的一个行组包括三个部分:

第一部分是行组头部的【同步标识】，主要用于分隔 hdfs 块中的两个连续行组

第二部分是行组的【元数据头部】，用于存储行组单元的信息，包括行组中的记录数、每个列的字节数、列中每个域的字节数

第三部分是【表格数据段】，即实际的列存储数据。在该部分中，同一列的所有域顺序存储

ORCFile:存储方式：数据按行分块 每块按照列存储,压缩快 快速列存取,效率比 rcfile 高,是 rcfile 的改良版本

52.【简答题】【Hive】Hive 的函数：UDF、UDAF、UDTF 的区别？

UDF:　单行进入，单行输出

UDAF:　多行进入，单行输出

UDTF:　单行输入，多行输出

53.【简答题】【Hive】简述 Hive 主要架构及解析成 MR 的过程？

Hive 元数据默认存储在 derby 数据库，不支持多客户端访问，所以需要将元数据存储在 MySQL 中，才支持多客户端访问。主要架构如下：

Hive 解析成 MR 的过程：Hive 通过给用户提供一系列交互接口，接收到用户的指令(sql 语句)，结合元数据(metastore)，经过 Driver 内的解析器，编译器，优化器，执行器转换成 mapreduce（将 sql 转换成抽象语法树 AST 的解析器，将 AST 编译成逻辑执行计划的编译器，在对逻辑执行计划进行优化的优化器，最后将逻辑执行计划转换成 mapreduce），提交给 hadoop 中执行，最后将执行返回的结果输出到用户交互接口

54.【简答题】【Hive】Hive 导入数据的五种方式？

Load 方式，可以从本地或 HDFS 上导入，本地是 copy，HDFS 是移动

Insert 方式，往表里插入

As select 方式，根据查询结果创建表并插入数据

Location 方式，创建表并指定数据的路径

Import 方式，先从 hive 上使用 export 导出在导入

55.【简答题】【Hive】Hive 中 4 种排序的区别？

Sort By：分区内有序

Order By：全局排序，只有一个 Reducer

Distrbute By：类似 MR 中 Partition，进行分区，结合 sort by 使用

Cluster By：当 Distribute by 和 Sorts by 字段相同时，可以使用 Cluster by 方式。Cluster by 除了具有 Distribute by 的功能外还兼具 Sort by 的功能。但是排序只能是升序排序，不能指定排序规则为 ASC 或者 DESC

56.【简答题】【Hive】简述 Hive 常见的数据导出方式。

Hive 的常见的数据导出方式有：

（1）从本地文件系统中导入数据到 Hive 表；

```
load data local inpath '/opt/data/test.txt' into table user;
```

（2）从 HDFS 上导入数据到 Hive 表；

```
load data inpath '/iflytek/test.txt' into table user;
```

（3）从别的表中查询出相应的数据并导入到 Hive 表中；

```
insert into table test   partition (age='25')   id, name, tel> from user;
```

（4）在创建表的时候通过从别的表中查询出相应的记录并插入到所创建的表中。

create table test2 as select id, name from user;

（5）Hive 表导出到本地文件系统

```
insert overwrite local directory '/iflysse/data/output' ROW FORMAT DELIMITED FIELDS TERMINATED by
',' select * from testA;
```

（6）Hive 表导出到 HDFS

```
INSERT OVERWRITE DIRECTORY '/iflysse/data/output1' select * from testA;
```

57.【简答题】【Spark】在使用 Spark 并行计算时，经常会出现“数据倾斜”问题，该问题的原因是什么，如何避免？

数据倾斜的原因：将数据分配给不同的任务一般是在 Shuffle 过程中完成的，在 Shuffle 操作中，键相同的数据一般会被分配到同一分区上并交给某个任务处理，当某个键的数量太大时（远远大于其他键的数量），就会导致数据倾斜，如图所示。数据倾斜最致命的问题就是内存溢出（OOM）。

避免措施：

对源数据进行聚合并过滤导致数据倾斜的键：

根据键对数据进行统计，找出导致数据倾斜的键。如果导致数据倾斜的键不影响业务的计算，就将这些键（例如−1、null 等）过滤掉，这种情况可能导致数据倾斜。另外，诸如−1 或 null 这样的键是“脏数据”，使用 filter 算子可将这些导致数据倾斜的键过滤掉，这样其他的键就有了均衡的值，从而消除了数据倾斜。

使用随机数消除数据倾斜：

当导致数据倾斜的键对应的数据为有效数据时，我们就不能简单地对数据进行过滤了，可以通过为倾斜的键加上随机数来消除数据倾斜。过程如下：针对键执行 map()函数，加上随机数，计算出结果后，再次针对键执行 map()函数，将随机数去掉，这样计算得到的结果和不加随机数时计算得到的结果是一样的，但好处在于可以将数据均等分配到多个任务中并进行计算。

58.【简答题】【Spark】Spark 常用算子 reduceByKey 与 groupByKey 的区别，哪一种更具优势?

reduceByKey:按照 key 进行聚合，在 shuffle 之前有 combine(预聚合)操作，返回结果是 RDD[k,v]。

groupByKey:按照 key 进行分组，直接进行 shuffle。

开发指导:reduceByKey 比 groupByKey 建议使用。但是需要注意是否会影响业务逻辑。

59.【简答题】【Spark】如何使用 Spark 实现 topN 的获取(描述思路或使用伪代码)

方法 1:

(1)按照 key 对数据进行聚合(groupByKey)

(2)将 value 转换为数组，利用 scala 的 sortBy 或者 sortWith 进行排序(mapValues)数据量太大，会 OOM。

方法 2:

(1)取出所有的 key

(2)对 key 进行迭代，每次取出一个 key 利用 spark 的排序算子进行排序

方法 3:

(1)自定义分区器，按照 key 进行分区，使不同的 key 进到不同的分区

(2)对每个分区运用 spark 的排序算子进行排序

60.【简答题】【Spark】Spark 为什么比 MapReduce 块？

1）基于内存计算，减少低效的磁盘交互；

2）高效的调度算法，基于 DAG；

3）容错机制 Linage，精华部分就是 DAG 和 Lingae

61.【简答题】【Spark】Spark 的优化怎么做？

Spark 调优比较复杂，但是大体可以分为三个方面来进行

1）平台层面的调优：防止不必要的 jar 包分发，提高数据的本地性，选择高效的存储格式如 parquet

2）应用程序层面的调优：过滤操作符的优化降低过多小任务，降低单条记录的资源开销，处理数据倾斜，复用 RDD 进行缓存，作业并行化执行等等

3）JVM 层面的调优：设置合适的资源量，设置合理的 JVM，启用高效的序列化方法如 kyro，增大 offhead 内存等等

62.【简答题】【Spark】RDD 有哪些缺陷？

1）不支持细粒度的写和更新操作（如网络爬虫），spark 写数据是粗粒度的。所谓粗粒度，就是批量写入数据，为了提高效率。但是读数据是细粒度的也就是说可以一条条的读；

2）不支持增量迭代计算，Flink 支持。

63.【简答题】【Spark】Spark 为什么要持久化，一般什么场景下要进行 persist 操作？

为什么要进行持久化？

spark 所有复杂一点的算法都会有 persist 身影，spark 默认数据放在内存，spark 很多内容都是放在内存的，非常适合高速迭代，1000 个步骤只有第一个输入数据，中间不产生临时数据，但分布式系统风险很高，所以容易出错，就要容错，rdd 出错或者分片可以根据血统算出来，如果没有对父 rdd 进行 persist 或者 cache 的化，就需要重头做。

以下场景会使用 persist：

1）某个步骤计算非常耗时，需要进行 persist 持久化；

2）计算链条非常长，重新恢复要算很多步骤；

3）checkpoint 所在的 rdd 要持久化 persist。checkpoint 前，要持久化，写个 rdD、cache 或者 rdD、persist，将结果保存起来，再写 checkpoint 操作，这样执行起来会非常快，不需要重新计算 rdd 链条了。checkpoint 之前一定会进行 persist；

4）shuffle 之后要 persist，shuffle 要进性网络传输，风险很大，数据丢失重来，恢复代价很大；

5）shuffle 之前进行 persist，框架默认将数据持久化到磁盘，这个是框架自动做的。

64.【简答题】【Spark】介绍一下 join 操作优化经验？

join 其实常见的就分为两类：map-sidejoin 和 reduce-sidejoin。当大表和小表 join 时，用 map-sidejoin 能显著提高效率。将多份数据进行关联是数据处理过程中非常普遍的用法，不过在分布式计算系统中，这个问题往往会变的非常麻烦，因为框架提供的 join 操作一般会将所有数据根据 key 发送到所有的 reduce 分区中去，也就是 shuffle 的过程。造成大量的网络以及磁盘 IO 消耗，运行效率极其低下，这个过程一般被称为 reduce-side-join。如果其中有张表较小的话，我们则可以自己实现在 map 端实现数据关联，跳过大量数据进行 shuffle 的过程，运行时间得到大量缩短，根据不同数据可能会有几倍到数十倍的性能提升。

65.【简答题】【Spark】不需要排序的 hashshuffle 是否一定比需要排序的 sortshuffle 速度快？

不一定，当数据规模小，Hashshuffle 快于 SortedShuffle 数据规模大的时候；当数据量大，sortedShuffle 会比 Hashshuffle 快很多，因为数量大的有很多小文件，不均匀，甚至出现数据倾斜，消耗内存大，1.x 之前 spark 使用 hash，适合处理中小规模，1.x 之后，增加了 Sortedshuffle，Spark 更能胜任大规模处理了。

66.【简答题】【Spark】Spark 并行度怎么设置比较合适？

spark 并行度，每个 core 承载 2~4 个 partition,如，32 个 core，那么 64~128 之间的并行度，也就是设置 64~128 个 partion，并行读和数据规模无关，只和内存使用量和 cpu 使用时间有关。

67.【简答题】【Spark】Spark on Mesos 中，什么是的粗粒度分配，什么是细粒度分配，各自的优点和缺点是什么？

1）粗粒度：启动时就分配好资源， 程序启动，后续具体使用就使用分配好的资源，不需要再分配资源。

好处：作业特别多时，资源复用率高，适合粗粒度。

坏处：容易资源浪费，假如一个 job 有 1000 个 task，完成了 999 个，还有一个没完成，那么使用粗粒度，999

个资源就会闲置在那里，资源浪费。

2）细粒度分配：用资源的时候分配，用完了就立即回收资源，启动会麻烦一点，启动一次分配一次，会比较麻烦。

68.【简答题】【Spark】spark 如何保证宕机迅速恢复?

1）适当增加 spark standby master

2）编写 shell 脚本，定期检测 master 状态，出现宕机后对 master 进行重启操作

69.【简答题】【Spark】Spark streaming 以及基本工作原理？

Spark streaming 是 spark core API 的一种扩展，可以用于进行大规模、高吞吐量、容错的实时数据流的处理。它支持从多种数据源读取数据，比如 Kafka、Flume、Twitter 和 TCP Socket，并且能够使用算子比如 map、reduce、join 和 window 等来处理数据，处理后的数据可以保存到文件系统、数据库等存储中。

Spark streaming 内部的基本工作原理是：接受实时输入数据流，然后将数据拆分成 batch，比如每收集一秒的数据封装成一个 batch，然后将每个 batch 交给 spark 的计算引擎进行处理，最后会生产处一个结果数据流，其中的数据也是一个一个的 batch 组成的。

70.【简答题】【Spark】spark 有哪些组件？

master：管理集群和节点，不参与计算。

worker：计算节点，进程本身不参与计算，和 master 汇报。Driver：运行程序的 main 方法，创建 spark context 对象。

spark context：控制整个 application 的生命周期，包括 dagsheduler 和 task scheduler 等组件。

client：用户提交程序的入口。

71.【简答题】【Spark】spark 工作机制？

用户在 client 端提交作业后，会由 Driver 运行 main 方法并创建 spark context 上下文。执行 add 算子，形成 dag 图输入 dagscheduler，按照 add 之间的依赖关系划分 stage 输入 task scheduler。task scheduler 会将 stage 划分为 task set 分发到各个节点的 executor 中执行。

72.【简答题】【Spark】Spark 有几种部署方式?请分别简要论述

1)Local:运行在一台机器上，通常是练手或者测试环境。

2)Standalone:构建一个基于 Master+Slaves 的资源调度集群，Spark 任务提交给 Master 运行。是 Spark 自身的一个调度系统。

3)Yarn:Spark 客户端直接连接 Yarn，不需要额外构建 Spark 集群。有 yarn-client 和 yarn-cluster 两种模式，主要区别在于:Driver 程序的运行节点。

4)Mesos:国内大环境比较少用。

73.【简答题】【Spark】Spark 的工作机制?

用户在 client 端提交作业后，会由 Driver 运行 main 方法并创建 sparkcontext 上下文。执行 add 算子，形成 dag 图输入 dagscheduler，按照 add 之间的依赖关系划分 stage 输入 taskscheduler。taskscheduler 会将 stage 划分为 taskset 分发到各个节点的 executor 中执行。

74.【简答题】【Spark】简述 Spark 的宽窄依赖，以及 Spark 如何划分 stage，每个 stage 又根据什么决定 task 个数?

Stage:根据 RDD 之间的依赖关系的不同将 Job 划分成不同的 Stage，遇到一个宽依赖则划分一个 Stage。

Task:Stage 是一个 TaskSet，将 Stage 根据分区数划分成一个个的 Task。

75.【简答题】【Spark】请列举 Spark 的 transformation 算子(不少于 8 个)，并简述功能

1)map(func):返回一个新的 RDD，该 RDD 由每一个输入元素经过 func 函数转换后组成.

2)mapPartitions(func):类似于 map，但独立地在 RDD 的每一个分片上运行，因此在类型为 T

的 RD 上运行时，func 的函数类型必须是 Iterator[T]=>Iterator[U]。假设有 N 个元素，有 M 个分区，那么 map 的函数的将被调用 N 次,而 mapPartitions 被调用 M 次,一个函数一次处理所有分区。

3)reduceByKey(func，[numTask]):在一个(K,V)的 RDD 上调用，返回一个(K,V)的 RDD，使用定的 reduce 函数，将相同 key 的值聚合到一起，reduce 任务的个数可以通过第二个可选的参数来设置。

4)aggregateByKey(zeroValue:U,[partitioner:Partitioner])(seqOp:(U,V)=>U,combOp:(U,U)=>U:在 kv 对的 RDD 中，，按 key 将 value 进行分组合并，合并时，将每个 value 和初始值作为 seq 函数的参数，进行计算，返回的结果作为一个新的 kv 对，然后再将结果按照 key 进行合并，最后将每个分组的 value 传递给 combine 函数进行计算(先将前两个 value 进行计算，将返回结果和下一个 value 传给 combine 函数，以此类推)，将 key 与计算结果作为一个新的 kv 对输出。

5)combineByKey(createCombiner:V=>C,mergeValue:(C,V)=>C,mergeCombiners:(C,C)=>C):对相同 K，把 V 合并成一个集合。

76.【简答题】【Spark】简述 SparkSQL 中 RDD、DataFrame、DataSet 三者的区别与联系?

1)RDD 优点:

编译时类型安全：编译时就能检查出类型错误

面向对象的编程风格

直接通过类名点的方式来操作数据

缺点:

序列化和反序列化的性能开销

无论是集群间的通信,还是 IO 操作都需要对对象的结构和数据进行序列化和反序列化。GC 的性能开销，频繁的创建和销毁对象,势必会增加 GC

2)DataFrame

DataFrame 引入了 schema 和 off-heapschema:RDD 每一行的数据,结构都是一样的，这个结构就存储在 schema 中。Spark 通过 schema 就能够读懂数据,因此在通信和 IO 时就只需要序列化和反序列化数据,而结构的部分就可以省略了。

3)DataSet

DataSet 结合了 RDD 和 DataFrame 的优点，并带来的一个新的概念 Encoder。

当序列化数据时，Encoder 产生字节码与 off-heap 进行交互，能够达到按需访问数据的效果，而不用反序列化整个对象。Spark 还没有提供自定义 Encoder 的 API，但是未来会加入。

77.【简答题】【Spark】SparkStreaming 有哪几种方式消费 Kafka 中的数据，它们之间的区别是什么?

一、基于 Receiver 的方式

这种方式使用 Receiver 来获取数据。Receiver 是使用 Kafka 的高层次 ConsumerAPI 来实现的。receiver 从 Kafka 中获取的数据都是存储在 SparkExecutor 的内存中的(如果突然数据暴增，大量 batch 堆积，很容易出现内存溢出的问题)，然后 SparkStreaming 启动的 job 会去处理那些数据。然而，在默认的配置下，这种方式可能会因为底层的失败而丢失数据。如果要启用高可靠机制，让数据零丢失，就必须启用 SparkStreaming 的预写日志机制(WriteAheadLog，WAL)。

该机制会同步地将接收到的 Kafka 数据写入分布式文件系统(比如 HDFS)上的预写日志中。所以，即使底层节点出现了失败，也可以使用预写日志中的数据进行恢复。

二、基于 Direct 的方式

这种新的不基于 Receiver 的直接方式，是在 Spark1.3 中引入的，从而能够确保更加健壮的机制。替代掉使用 Receiver 来接收数据后，这种方式会周期性地查询 Kafka，来获得每个 topic+partition 的最新的 offset，从而定义每个 batch 的 offset 的范围。当处理数据的 job 启动时，就会使用 Kafka 的简单 consumerapi 来获取 Kafka 指定 offset 范围的数据。

优点如下:

简化并行读取:如果要读取多个 partition，不需要创建多个输入 DStream 然后对它们进行 union 操作。Spark 会创建跟 Kafkapartition 一样多的 RDDpartition，并且会并行从 Kafka 中读取数据。所以在 Kafkapartition 和 RDDpartition 之间，有一个一对一的映射关系。

高性能:如果要保证零数据丢失，在基于 receiver 的方式中，需要开启 WAL 机制。这种方式其实效率低下，因为数据实际上被复制了两份，Kafka 自己本身就有高可靠的机制，会对数据复制一份，而这里又会复制一份到 WAL 中。而基于 direct 的方式，不依赖 Receiver，不需要开启 WAL 机制，只要 Kafka 中作了数据的复制，那么就可以通过 Kafka 的副本进行恢复。一次且仅一次的事务机制。

三、对比:

基于 receiver 的方式，是使用 Kafka 的高阶 API 来在 ZooKeeper 中保存消费过的 offset 的。这是消费 Kafka 数据的传统方式。这种方式配合着 WAL 机制可以保证数据零丢失的高可靠性，但是却无法保证数据被处理一次且仅一次，可能会处理两次。因为 Spark 和 ZooKeeper 之间可能是不同步的。基于 direct 的方式，使用 kafka 的简单 api，SparkStreaming 自己就负责追踪消费的 offset，并保存在 checkpoint 中。Spark 自己一定是同步的，因此可以保证数据是消费一次且仅消费一次。在实际生产环境中大都用 Direct 方式

78.【简答题】【Spark】调优之前与调优之后性能的详细对比(例如调整 map 个数，map 个数之前多少、之后多少，有什么提升)

这里举个例子。比如我们有几百个文件，会有几百个 map 出现，读取之后进行 join 操作，会非常的慢。这个时候我们可以进行 coalesce 操作，比如 240 个 map，我们合成 60 个 map，也就是窄依赖。这样再 shuffle，过程产生的文件数会大大减少。提高 join 的时间性能。

79.【简答题】【HDFS】HDFS 在读取文件的时候,如果其中一个块突然损坏了怎么办?

客户端读取完 DataNode 上的块之后会进行 checksum 验证，也就是把客户端读取到本地的块与 HDFS 上的原始块进行校验，如果发现校验结果不一致，客户端会通知 NameNode，然后再从下一个拥有该 block 副本的 DataNode 继续读。

80.【简答题】【HDFS】请描述一下 HDFS 的读写流程。

HDFS 写流程：

1)client 客户端发送上传请求，通过 RPC 与 namenode 建立通信，namenode 检查该用户是否有上传权限，以及上传的文件是否在 hdfs 对应的目录下重名，如果这两者有任意一个不满足，则直接报错，如果两者都满足，则返回给客户端一个可以上传的信息

2)client 根据文件的大小进行切分，默认 128M 一块，切分完成之后给 namenode 发送请求第一个 block 块上传到哪些服务器上

3)namenode 收到请求之后，根据网络拓扑和机架感知以及副本机制进行文件分配，返回可用的 DataNode 的地址

注:Hadoop 在设计时考虑到数据的安全与高效,数据文件默认在 HDFS 上存放三份,存储策略为本地一份，同机架内其它某一节点上一份,不同机架的某一节点上一份

4)客户端收到地址之后与服务器地址列表中的一个节点如 A 进行通信，本质上就是 RPC 调用，建立 pipeline，A 收到请求后会继续调用 B，B 在调用 C，将整个 pipeline 建立完成，逐级返回 client

5)client 开始向 A 上发送第一个 block(先从磁盘读取数据然后放到本地内存缓存)，以 packet(数据包，64kb)为单位，A 收到一个 packet 就会发送给 B，然后 B 发送给 C，A 每传完一个 packet 就会放入一个应答队列等待应答

6)数据被分割成一个个的 packet 数据包在 pipeline 上依次传输，在 pipeline 反向传输中，逐个发送 ack(命令正确应答)，最终由 pipeline 中第一个 DataNode 节点 A 将 pipelineack 发送给 Client

7)当一个 block 传输完成之后,Client 再次请求 NameNode 上传第二个 block，namenode 重新选择三台 DataNode 给 client

HDFS 读流程：

1)client 向 namenode 发送 RPC 请求。请求文件 block 的位置；

2)namenode 收到请求之后会检查用户权限以及是否有这个文件，如果都符合，则会视情况返回部分或全部的 block 列表，对于每个 block，NameNode 都会返回含有该 block 副本的 DataNode 地址;这些返回的 DN 地址，会按照集群拓扑结构得出 DataNode 与客户端的距离，然后进行排序，排序两个规则:网络拓扑结构中距离 Client 近的排靠前;心跳机制中超时汇报的 DN 状态为 STALE，这样的排靠后；

3)Client 选取排序靠前的 DataNode 来读取 block，如果客户端本身就是 DataNode,那么将从本地直接获取数据(短路读取特性)；

4)底层上本质是建立 SocketStream(FSDataInputStream)，重复的调用父类 DataInputStream 的 read 方法，直到这个块上的数据读取完毕；

5)当读完列表的 block 后，若文件读取还没有结束，客户端会继续向 NameNode 获取下一批的 block 列表

6)读取完一个 block 都会进行 checksum 验证，如果读取 DataNode 时出现错误，客户端会通知 NameNode，然后再从下一个拥有该 block 副本的 DataNode 继续读；

7)read 方法是并行的读取 block 信息，不是一块一块的读取;NameNode 只是返回 Client 请求包含块的 DataNode 地址，并不是返回请求块的数据；

8)最终读取来所有的 block 会合并成一个完整的最终文件；

81.【简答题】【HDFS】HDFS 在上传文件的时候,如果其中一个 DataNode 突然挂掉了怎么办?

客户端上传文件时与 DataNode 建立 pipeline 管道，管道正向是客户端向 DataNode 发送的数据包，管道反向是 DataNode 向客户端发送 ack 确认，也就是正确接收到数据包之后发送一个已确认接收到的应答，当 DataNode 突然挂掉了，客户端接收不到这个 DataNode 发送的 ack 确认，客户端会通知 NameNode，NameNode 检查该块的副本与规定的不符，NameNode 会通知 DataNode 去复制副本，并将挂掉的 DataNode 作下线处理，不再让它参与文件上传与下载。

82.【简答题】【HDFS】小文件过多会有什么危害,如何避免

Hadoop 上大量 HDFS 元数据信息存储在 NameNode 内存中,因此过多的小文件必定会压垮 NameNode 的内存每个元数据对象约占 150byte，所以如果有 1 千万个小文件，每个文件占用一个 block，则 NameNode 大约需要 2G 空间。如果存储 1 亿个文件，则 NameNode 需要 20G 空间。显而易见的解决这个问题的方法就是合并小文件,可以选择在客户端上传时执行一定的策略先合并,或者是使用 Hadoop 的 CombineFileInputFormat<K,V>实现小文件的合并

83.【简答题】【HDFS】NameNode 出现故障时如何恢复？

需要执行以下步骤以使 Hadoop 集群正常运行：

使用文件系统元数据副本 FsImage 来启动新的 NameNode。

配置数据节点以及客户端，使它们确认新启动的名称节点。

一旦新的 NameNode 完成加载最后一个从 DataNode 接收到足够阻止报告的检查点 FsImage，它将开始为客户端提供服务。

对于大型 Hadoop 集群，NameNode 恢复过程会耗费大量时间，这对于例行维护来说是一个更大的挑战。

84.【简答题】【HDFS】为什么 HDFS 仅适用于大型数据集，而不适用于许多小型文件的正确工具？

这是由于 NameNode 的性能问题。通常，为 NameNode 分配了巨大的空间来存储大型文件的元数据。为了获得最佳的空间利用和成本效益，元数据应该来自单个文件。对于小文件，NameNode 不会利用整个空间，这是性能优化的问题。

85.【简答题】【HDFS】当两个用户尝试写入 HDFS 中的同一文件时会发生什么？

HDFS NameNode 仅支持独占写入。因此，只有第一个用户将获得文件访问许可，而第二个用户将被拒绝。

86.【单选题】【HDFS】 关于 Hadoop Shell 操作不正确的是（ ）

A、bin/Hadoop jar [-target.jar] 表示打包运行 MapReduce 程序

B、bin/Hadoop jar [-target.jar] 表示打包运行 MapReduce 程序

C、bin/Hadoop fs －put [-txt] / 表示从本地上传文件到 hdfs 上

D、bin/hdfs dfs －ls －R / 表示只查看 hdfs 上根目录下的目录与文件

答案：D。

解析：参数-R 是代表递归显示

87.【单选题】【HDFS】执行数据块复制的任务是，是下列选项中的哪两个在进行通信？（ ）

A、 client and namenode

B、 client and datanode

C、 namenode and datanode

D、 datanode and datanode

答案：D。

解析：执行数据块复制的任务是，是 datanode 和 datanode 在进行通信。

88.【单选题】【HDFS】HDFS 无法高效存储大量小文件，想让它能处理好小文件，比较可行的改进策略不包括（）

A、利用 SequenceFile、MapFile、Har 等方式归档小文件

B、多 Master 设计

C、Block 大小适当调小

D、调大 namenode 内存或将文件系统元数据存到硬盘里

答案：D。

解析：HDFS 的特性通过调大 namenode 内存或将文件系统元数据存到硬盘里并不能使 HDF 处理好小文件。

89.【单选题】【HDFS】下面选项中哪两个进程是分别负责管理 HDFS 数据存储和备份元数据信息的？（ ）

A、 NameNode、DataNode

B、 NameNode、Jobtracker

C、 Datanode，SecondaryNameNode

D、 NameNode、SecondaryNameNode

答案：D。

解析：NameNode 负责 HDFS 的 DataNode, SecondaryNameNode 备份 Namenode 的元数据信息等，DataNode 是真是存储数据的数据块。

90.【单选题】【HDFS】Client 在 HDFS 上进行文件写入时，namenode 根据文件大小和配置情况，返回部分 datanode 信息，（　）负责将文件划分为多个 Block，根据 DataNode 的地址信息，按顺序写入到每一个 DataNode 块。

A、Client

B、Namenode

C、Datanode

D、Secondary namenode

答案：A。

解析：HDFS 写入时，Client 只上传数据到一台 DataNode，然后由 NameNode 负责 Block 复制工作。

91.【单选题】【HDFS】 Hadoop 中下面哪项操作是不需要记录进日志的？（　）

A、打开文件

B、重命名

C、编译文件

D、删除操作

答案：C。

解析：编译文件是不需要记录进日志中的。

92.【简答题】【HDFS】NameNode 在启动的时候会做哪些操作？

NameNode 数据存储在内存和本地磁盘，本地磁盘数据存储在 fsimage 镜像文件和 edits 编辑日志文件

首次启动 NameNode：

1、格式化文件系统，为了生成 fsimage 镜像文件

2、启动 NameNode

(1)读取 fsimage 文件，将文件内容加载进内存

(2)等待 DataNade 注册与发送 blockreport

3、启动 DataNode

(1)向 NameNode 注册

(2)发送 blockreport

(3)检查 fsimage 中记录的块的数量和 blockreport 中的块的总数是否相同 4、对文件系统进行操作(创建目录，上传文件，删除文件等)(1)此时内存中已经有文件系统改变的信息，但是磁盘中没有文件系统改变的信息，此时会将这些改变信息写入 edits 文件中，edits 文件中存储的是文件系统元数据改变的信息。

第二次启动 NameNode：

1、读取 fsimage 和 edits 文件

2、将 fsimage 和 edits 文件合并成新的 fsimage 文件

3、创建新的 edits 文件，内容为空

4、启动 DataNode

93.【简答题】【HDFS】SecondaryNameNode 了解吗，它的工作机制是怎样的？

SecondaryNameNode 是合并 NameNode 的 editlogs 到 fsimage 文件中;

它的具体工作机制:

(1)SecondaryNameNode 询问 NameNode 是否需要 checkpoint。直接带回 NameNode 是否检查结果

(2)SecondaryNameNode 请求执行 checkpoint

(3)NameNode 滚动正在写的 edits 日志

(4)将滚动前的编辑日志和镜像文件拷贝到 SecondaryNameNode

(5)SecondaryNameNode 加载编辑日志和镜像文件到内存，并合并

(6)生成新的镜像文件 fsimage.chkpoint

(7)拷贝 fsimage.chkpoint 到 NameNode

(8)NameNode 将 fsimage.chkpoint 重新命名成 fsimage，所以如果 NameNode 中的元数据丢失，是可以从 SecondaryNameNode 恢复一部分元数据信息的，但不是全部，因为 NameNode 正在写的 edits 日志还没有拷贝到 SecondaryNameNode，这部分恢复不了

94.【简答题】【HDFS】HDFS 存入大量小文件，有什么影响?

元数据层面:每个小文件都有一份元数据，其中包括文件路径，文件名，所有者，所属组，权限，创建时间等，这些信息都保存在 Namenode 内存中。所以小文件过多，会占用 Namenode 服务器大量内存，影响 Namenode 性能和使用寿命

计算层面:默认情况下 MR 会对每个小文件启用一个 Map 任务计算，非常影响计算性能。同时也影响磁盘寻址时间。

95.【简答题】【HDFS】请说下 HDFS 的组织架构

1)Client:客户端

(1)切分文件。文件上传 HDFS 的时候，Client 将文件切分成一个一个的 Block，然后进行存储

(2)与 NameNode 交互，获取文件的位置信息

(3)与 DataNode 交互，读取或者写入数据

(4)Client 提供一些命令来管理 HDFS，比如启动关闭 HDFS、访问 HDFS 目录及内容等

2)NameNode:名称节点，也称主节点，存储数据的元数据信息，不存储具体的数据

(1)管理 HDFS 的名称空间

(2)管理数据块(Block)映射信息

(3)配置副本策略

(4)处理客户端读写请求

3)DataNode:数据节点，也称从节点。NameNode 下达命令，DataNode 执行实际的操作

(1)存储实际的数据块

(2)执行数据块的读/写操作

4)SecondaryNameNode:并非 NameNode 的热备。当 NameNode 挂掉的时候，它并不能马上替换 NameNode 并提供服务

(1)辅助 NameNode，分担其工作量

(2)定期合并 Fsimage 和 Edits，并推送给 NameNode

(3)在紧急情况下，可辅助恢复 NameNode

96.【简答题】【HDFS】请说下 HDFS 读写流程

HDFS 写流程

1）client 客户端发送上传请求，通过 RPC 与 namenode 建立通信，namenode 检查该用户是否有上传权限，以及上传的文件是否在 hdfs 对应的目录下重名，如果这两者有任意一个不满足，则直接报错，如果两者都满足，则返回给客户端一个可以上传的信息

2）client 根据文件的大小进行切分，默认 128M 一块，切分完成之后给 namenode 发送请求第一个 block 块上传到哪些服务器上

3）namenode 收到请求之后，根据网络拓扑和机架感知以及副本机制进行文件分配，返回可用的 DataNode 的地址

4）客户端收到地址之后与服务器地址列表中的一个节点如 A 进行通信，本质上就是 RPC 调用，建立 pipeline，A 收到请求后会继续调用 B，B 在调用 C，将整个 pipeline 建立完成，逐级返回 client

5）client 开始向 A 上发送第一个 block（先从磁盘读取数据然后放到本地内存缓存），以 packet（数据包，64kb）为单位，A 收到一个 packet 就会发送给 B，然后 B 发送给 C，A 每传完一个 packet 就会放入一个应答队列等待应答

6）数据被分割成一个个的 packet 数据包在 pipeline 上依次传输，在 pipeline 反向传输中，逐个发送 ack（命令正确应答），最终由 pipeline 中第一个 DataNode 节点 A 将 pipelineack 发送给 Client

7）当一个 block 传输完成之后, Client 再次请求 NameNode 上传第二个 block ，namenode 重新选择三台 DataNode 给 client

HDFS 读流程

1）client 向 namenode 发送 RPC 请求。请求文件 block 的位置

2）namenode 收到请求之后会检查用户权限以及是否有这个文件，如果都符合，则会视情况返回部分或全部的 block 列表，对于每个 block，NameNode 都会返回含有该 block 副本的 DataNode 地址； 这些返回的 DN 地址，会按照集群拓扑结构得出 DataNode 与客户端的距离，然后进行排序，排序两个规则：网络拓扑结构中距离

Client 近的排靠前；心跳机制中超时汇报的 DN 状态为 STALE，这样的排靠后

3）Client 选取排序靠前的 DataNode 来读取 block，如果客户端本身就是 DataNode,那么将从本地直接获取数据(短路读取特性) 4）底层上本质是建立 Socket Stream（FSDataInputStream），重复的调用父类 DataInputStream 的 read 方法，直到这个块上的数据读取完毕

5）当读完列表的 block 后，若文件读取还没有结束，客户端会继续向 NameNode 获取下一批的 block 列表

6）读取完一个 block 都会进行 checksum 验证，如果读取 DataNode 时出现错误，客户端会通知 NameNode，然后再从下一个拥有该 block 副本的 DataNode 继续读

7）read 方法是并行的读取 block 信息，不是一块一块的读取；NameNode 只是返回 Client 请求包含块的 DataNode 地址，并不是返回请求块的数据

8） 最终读取来所有的 block 会合并成一个完整的最终文件

97.【单选题】【HDFS】下列哪项通常是集群的最主要瓶颈？（ ）

A、CPU

B、网络

C、磁盘 IO

D、内存

答案：C。

解析：集群的最主要瓶颈是磁盘 IO。

98.【简答题】【HDFS】什么是 fsck？

fsck 代表文件系统检查。这是 HDFS 使用的命令。此命令用于检查不一致以及文件中是否存在任何问题。例如，如果文件缺少任何块，则 HDFS 将通过此命令得到通知。

99.【简答题】【HDFS】NAS（网络附加存储）和 HDFS 之间的主要区别是什么？

NAS（网络附加存储）和 HDFS 之间的主要区别 -

HDFS 在计算机集群上运行，而 NAS 在单台计算机上运行。因此，数据冗余是 HDFS 中的常见问题。相反，对于 NAS，复制协议是不同的。因此，数据冗余的机会要少得多。

对于 HDFS，数据将作为数据块存储在本地驱动器中。对于 NAS，它存储在专用硬件中。

100.【简答题】【HDFS】格式化 NameNode 的命令是什么？

$ hdfs namenode -format

101.【简答题】【HDFS】“ HDFS 块”和“输入分割”之间有什么区别？

HDFS 将输入数据物理上划分为块进行处理，这称为 HDFS 块。

输入拆分是映射器对数据的逻辑划分，用于映射操作。

102.【简答题】【HDFS】NFS 与 HDFS 有何不同？

有许多分布式文件系统以它们自己的方式工作。NFS（网络文件系统）是最古老和流行的分布式文件存储系统之一，而 HDFS（Hadoop 分布式文件系统）是最近使用和流行的处理大数据的系统。 NFS 和 HDFS 之间的主要区别如下：

Criteria	NFS	HDFS
Data Size Support	NFS can store and process small amount of data	HDFS is mainly use to store and process big data
Data Storage	Data is stored on a single dedicated hardware	The data blocks are distributed on the local drives of hardware
Reliability	No reliability, data is not available in case of machine failure	Data is stored reliably, data is available even after machine failure
Data Redundancy	NFS runs on single machine, no chances of data redundancy	HDFS runs on a cluster of different machines, data redundancy may occur due to replication protocol

103.【简答题】【HDFS】NameNode，Task Tracker 和 Job Tracker 的端口号是什么？

NameNode - 端口 50070

任务跟踪器 - 端口 50060

作业跟踪器 - 端口 50030

104.【简答题】【HDFS】说明覆盖 HDFS 中复制因子的过程。

有两种方法可以覆盖 HDFS 中的复制因子

方法 1：基于文件

在此方法中，使用 Hadoop FS Shell 根据文件更改复制因子。用于此的命令是：

$ hadoop fs - setrep - w2 / my / test_file

在这里，test_file 是复制因子将设置为 2 的文件名。

方法 2：基于目录

在这种方法中，复制因子将基于目录进行更改，即，修改给定目录下所有文件的复制因子。

$ hadoop fs - setrep - w5 / my / test_dir

在这里，test_dir 是目录的名称，该目录及其中所有文件的复制因子将设置为 5。

105.【简答题】【HDFS】没有任何数据的 NameNode 会发生什么？

没有任何数据的 NameNode 在 Hadoop 中不存在。如果存在 NameNode，它将包含一些数据，否则将不存在。

106.【简答题】【HDFS】说明 NameNode 恢复过程。

NameNode 恢复过程涉及以下使 Hadoop 集群运行的步骤：

在恢复过程的第一步中，文件系统元数据副本（FsImage）启动一个新的 NameNode。

下一步是配置数据节点和客户端。然后，这些 DataNode 和客户端将确认新的 NameNode。

在最后一步中，新的 NameNode 在最后一个检查点 FsImage 加载完成并从 DataNode 接收块报告后开始为客户端提供服务。

注意：别忘了，在大型 Hadoop 集群上，此 NameNode 恢复过程会消耗大量时间。因此，这使得日常维护变得困难。因此，建议使用 HDFS 高可用性体系结构。

107.【简答题】【HDFS】DFS 可以处理大量数据，那么为什么我们需要 Hadoop 框架？

Hadoop 不仅用于存储大数据，而且还用于处理这些大数据。虽然 DFS（分布式文件系统）也可以存储数据，但是它缺乏以下功能：

-它不是容错的

-网络上的数据移动取决于带宽。

108.【简答题】【HDFS】HDFS 中的块是什么，在 Hadoop 1 和 Hadoop 2 中 其默认大小是多少？我们可以更改块大小吗？

块是硬盘中最小的连续数据存储。对于 HDFS，块跨 Hadoop 群集存储。

Hadoop 1 中的默认块大小为：64 MB

Hadoop 2 中的默认块大小为：128 MB

是的，我们可以使用 hdfs-site.xml 文件中的参数 dfs.block.size 更改块大小。

109.【简答题】【zookeeper】zk 节点宕机如何处理？

Zookeeper 本身也是集群，推荐配置不少于 3 个服务器。Zookeeper 自身也要保证当一个节点宕机时，其他节点会继续提供服务。

如果是一个 Follower 宕机，还有 2 台服务器提供访问，因为 Zookeeper 上的数据是有多个副本的，数据并不会丢失；

如果是一个 Leader 宕机，Zookeeper 会选举出新的 Leader。

ZK 集群的机制是只要超过半数的节点正常，集群就能正常提供服务。只有在 ZK 节点挂得太多，只剩一半或不到一半节点能工作，集群才失效。

因此 3 个节点的 cluster 可以挂掉 1 个节点(leader 可以得到 2 票>1.5)，2 个节点的 cluster 就不能挂掉任何 1 个节点了(leader 可以得到 1 票<=1)

110.【简答题】【zookeeper】zookeeper 简单介绍一下，为什么要用 zk？

Zookeeper 是一个分布式协调服务的开源框架。主要用来解决分布式集群中应用系统的一致性问题，例如怎样避免同时操作同一数据造成脏读的问题。

ZooKeeper 本质上是一个分布式的小文件存储系统。提供基于类似于文件系统的目录树方式的数据存储，并且可以对树中的节点进行有效管理。从而用来维护和监控你存储的数据的状态变化。通过监控这些数据状态的变化，从而可以达到基于数据的集群管理。诸如：统一命名服务(dubbo)、分布式配置管理(solr 的配置集中管理)、分布式消息队列（sub/pub）、分布式锁、分布式协调等功能。

111.【多选题】【Zookeeper】关于 Zookeeper 的数据结构，以下说法正确的有（）？

A、 与 Unix 文件系统类似

B、 整体可以看做成一棵树

C、 每个节点称为一个 Znode

D、 每个 Znode 默认存储 128MB 数据

答案：A，B、C。

Znode 默认储存为 1M

112.【简答题】【Hadoop】Hadoop 宕机如何解决？

1)如果 MR 造成系统宕机。此时要控制 Yarn 同时运行的任务数，和每个任务申请的最大内存。调整参数:yarn.scheduler.maximum-allocation-mb(单个任务可申请的最多物理内存量，默认是 8192MB)

2)如果写入文件过量造成 NameNode 宕机。那么调高 Kafka 的存储大小，控制从 Kafka 到 HDFS 的写入速度。高峰期的时候用 Kafka 进行缓存，高峰期过去数据同步会自动跟上。

113.【简答题】【Hadoop】如何重新启动 Hadoop 中的所有守护程序？

要重新启动所有守护程序，需要首先停止所有守护程序。Hadoop 目录包含 sbin 目录，该目录存储脚本文件以在 Hadoop 中停止和启动守护程序。

使用 stop daemons 命令/sbin/stop-all.sh 停止所有守护程序，然后使用/sin/start-all.sh 命令再次启动所有守护程

序。

114.【简答题】【hadoop】简述如何安装配置 apache 的一个开源的 hadoop

1.使用 root 账户登陆

2.修改 ip

3.修改 host 主机名

4.配置 ssh 免密登陆

5.关闭防火墙

6.安装ＪＤＫ

7.解压 hadoop 安装包

8.配置 hadoop 的核心配置文件 hadoop-env.sh? core-site.xml? mapred-site.xml yarn-site.xml hdfs-site.xml

9.配置 hadoop 的环境变量

10.格式化 hadoop namenode-format

11.启动节点 start-all.sh

115.【单选题】【Hadoop】关于服务启动正确的是（ ）

A、sbin/start-hdfs.sh

B、sbin/hadoop-daemon.sh start resourcemanager

C、sbin/hadoop-daemon.sh start-yarn.sh

D、sbin/hadoop-daemon.sh namenode start

答案：B。

解析：三种方式启动 Hadoop：

sbin/start-all.sh;

sbin/start-dfs.sh|start-yarn.sh；

sbin/hadoop-daemon.sh start XXX。

116.【单选题】【Hadoop】在安装配置好 Hadoop 集群后，查看 Namenode 节点的端口是以下哪个？（ ）

A、50030

B、50070

C、60010

D、60030

答案：B。

解析：NameNode 端口是 50070。

117.【单选题】【Hadoop】关于 Hadoop SSH 免密码登录说法错误的是（ ）

A、在 SSH 安全协议的原理中， 是一种非对称加密与对称加密算法的结合

B、生成公钥和私钥的只要输入~/.ssh/ssh-keygen –t rsa 命令，四个回车即可

C、生成 SSH 密钥之后，只需要输入 ssh-copy-id localhost 即可免密码登陆

D、在本机搭建好 SSH 免密码登陆，就可以随意的访问其它节点服务

答案：D。

解析：在本机搭建好 SSH 免密码登陆，只可以随意的访问本机的服务。

118.【简答题】【Hadoop】Yarn 的默认调度器、调度器分类、以及他们之间的区别？

Hadoop 调度器重要分为三类：FIFO 、Capacity Scheduler（容量调度器）和 Fair Sceduler（公平调度器）

FIFO 调度器：先进先出，同一时间队列中只有一个任务在执行

容量调度器：多队列；每个队列内部先进先出，同一时间队列中只有一个任务在执行。队列的并行度为队列的个数

公平调度器：多队列；每个队列内部按照缺额大小分配资源启动任务，同一时间队列中有多个任务执行。队列的并行度大于等于队列的个数

一定要强调生产环境中不是使用的 FifoScheduler，面试的时侯会发现候选人大概了解这几种调度器的区别，但是问在生产环境用哪种，却说使用的 FifoScheduler（企业生产环境一定不会用这个调度的）

119.【简答题】【Hadoop】LZO 压缩如何部署 Hadoop？

Hadoop 默认不支持 LZO 压缩，如果需要支持 LZO 压缩，需要添加 jar 包，并在 hadoop 的 cores-site.xml 文件中添加相关压缩配置

120.【简答题】【Hadoop】如果 Hadoop 宕机如何处理？

如果 MR 造成系统宕机。此时要控制 Yarn 同时运行的任务数，和每个任务申请的最大内存。调整参数：yarn.scheduler.maximum-allocation-mb（单个任务可申请的最多物理内存量，默认是 8192MB）

如果写入文件过量造成 NameNode 宕机。那么调高 Kafka 的存储大小，控制从 Kafka 到 HDFS 的写入速度。高峰期的时候用 Kafka 进行缓存，高峰期过去数据同步会自动跟上

121.【简答题】【Hadoop】Hadoop 的参数调优有哪些？

在 hdfs-site.xml 文件中配置多目录，最好提前配置好，否则更改目录需要重新启动集群；

NameNode 有一个工作线程池，用来处理不同 DataNode 的并发心跳以及客户端并发的元数据操作，dfs.namenode.handler.count=20 * log2(Cluster Size)，比如集群规模为 10 台时，此参数设置为 60；

编辑日志存储路径 dfs.namenode.edits.dir 设置与镜像文件存储路径 dfs.namenode.name.dir 尽量分开，达到最低写入延迟；

服务器节点上 YARN 可使用的物理内存总量，默认是 8192（MB），注意，如果你的节点内存资源不够 8GB，则需要调减小这个值，而 YARN 不会智能的探测节点的物理内存总量。yarn.nodemanager.resource.memory-mb

单个任务可申请的最多物理内存量，默认是 8192（MB）。yarn.scheduler.maximum-allocation-mb

122.【简答题】【Hadoop】哪种硬件配置最适合 Hadoop 作业？

配置 4/8 GB RAM 和 ECC 内存的双处理器或核心计算机是运行 Hadoop 操作的理想选择。但是，硬件配置会根据特定于项目的工作流和处理流程而有所不同，因此需要进行相应的自定义。

123.【简答题】【Hadoop】您对 Hadoop 中的 Rack Awareness 了解什么？

这是应用于 NameNode 的算法，用于确定如何放置块及其副本。根据机架定义，在同一机架内的 DataNode 之间将网络流量最小化。例如，如果我们将复制因子设为 3，则将两个副本放在一个机架中，而将第三副本放在一个单独的机架中。

124.【简答题】【Hadoop】解释 Hadoop 和 RDBMS 之间的区别。

Hadoop 和 RDBMS 之间的区别如下

Criteria	Hadoop	RDBMS
Schema	Based on 'Schema on Read'	Based on 'Schema on Write'
Data Types	Structured, Semi-structured and Unstructured data	Structured Data
Speed	Writes are Fast	Reads are Fast
Cost	Open source framework, free of cost	Licensed software, paid
Applications	Data discovery, Storage and processing of unstructured data	OLTP and complex ACID transactions

125.【简答题】【Hadoop】Hadoop 中常见的输入格式是什么？

以下是 Hadoop 中常见的输入格式 -

文本输入格式 – Hadoop 中定义的默认输入格式是文本输入格式。

序列文件输入格式 – 要读取序列中的文件，请使用序列文件输入格式。

键值输入格式 – 用于纯文本文件（分成几行的文件）的输入格式是键值输入格式。

126.【简答题】【Hadoop】解释 Hadoop 的一些重要特性。

Hadoop 支持大数据的存储和处理。它是应对大数据挑战的最佳解决方案。Hadoop 的一些重要功能是：

开源 – Hadoop 是一个开源框架，这意味着它是免费提供的。同样，允许用户根据他们的要求更改源代码。

分布式处理 – Hadoop 支持数据的分布式处理，即更快的处理。Hadoop HDFS 中的数据以分布式方式存储，而 MapReduce 负责数据的并行处理。

容错 – Hadoop 具有高度的容错能力。默认情况下，它将为每个块在不同节点上创建三个副本。该编号可以根据需要进行更改。因此，如果一个节点发生故障，我们可以从另一节点恢复数据。节点故障的检测和数据恢复是自动完成的。

可靠性 – Hadoop 以可靠的方式将数据存储在群集上，而与计算机无关。因此，存储在 Hadoop 环境中的数据不受计算机故障的影响。

可伸缩性 – Hadoop 的另一个重要功能是可伸缩性。它与其他硬件兼容，我们可以轻松地将新硬件装配到节点上。

高可用性 – 即使在硬件出现故障之后，也可以访问存储在 Hadoop 中的数据。如果发生硬件故障，可以从其他路径访问数据。

127.【简答题】【Hadoop】解释 Hadoop 运行的不同模式。

Apache Hadoop 在以下三种模式下运行 –

独立（本地）模式 – 默认情况下，Hadoop 以本地模式运行，即在非分布式单节点上运行。此模式使用本地文件系统执行输入和输出操作。此模式不支持使用 HDFS，因此用于调试。在此模式下，配置文件不需要自定义配置。

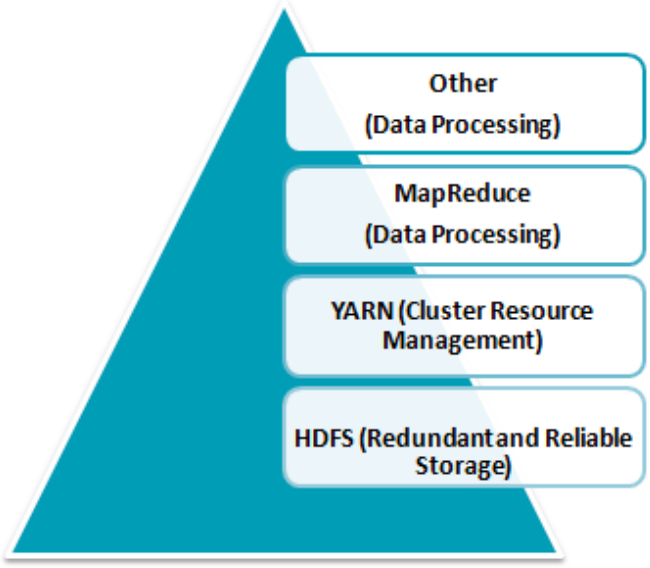
伪分布式模式 – 在伪分布式模式下，Hadoop 与独立模式一样在单个节点上运行。在这种模式下，每个守护程序都在单独的 Java 进程中运行。由于所有守护程序都在单个节点上运行，因此主节点和从节点都存在相同的节点。

完全分布式模式 – 在完全分布式模式下，所有守护程序都在单独的单个节点上运行，因此形成了多节点集群。主节点和从节点有不同的节点。

128.【简答题】【Hadoop】解释 Hadoop 的核心组件。

Hadoop 是一个开源框架，旨在以分布式方式存储和处理大数据。Hadoop 的核心组件是 –

HDFS（Hadoop 分布式文件系统） – HDFS 是 Hadoop 的基本存储系统。在商用硬件群集上运行的大型数据文件存储在 HDFS 中。即使硬件出现故障，它也可以以可靠的方式存储数据。



Hadoop的核心组件

Hadoop MapReduce – MapReduce 是负责数据处理的 Hadoop 层。它编写一个应用程序来处理存储在 HDFS 中的非结构化和结构化数据。通过将数据划分为独立的任务，它负责并行处理大量数据。该处理过程分为 Map 和 Reduce 两个阶段。映射是指定复杂逻辑代码的处理的第一阶段，而精简是指定轻量级操作的处理的第二阶段。

YARN – Hadoop 中的处理框架是 YARN。它用于资源管理，并提供多个数据处理引擎，即数据科学，实时流和批处理。

129.【简答题】【Hadoop】Hadoop CLASSPATH 对启动或停止 Hadoop 守护程序有何必要？

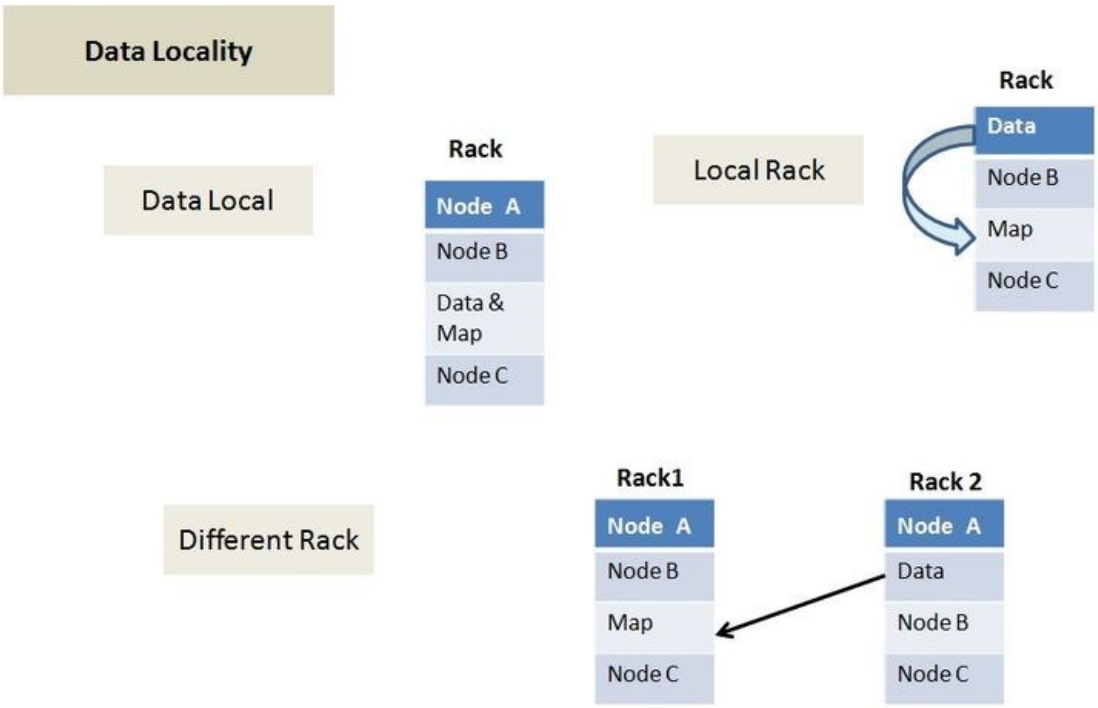
CLASSPATH 包含必要的目录，其中包含用于启动或停止 Hadoop 守护程序的 jar 文件。因此，设置 CLASSPATH 对于启动或停止 Hadoop 守护程序至关重要。

但是，每次设置 CLASSPATH 都不是我们遵循的标准。通常，CLASSPATH 是写在/etc/hadoop/hadoop-env.sh 文件中的。因此，一旦我们运行 Hadoop，它将自动加载 CLASSPATH。

130.【简答题】【Hadoop】为什么我们需要 Hadoop 中的数据局部性？说明。

HDFS 中的数据集在 Hadoop 集群的 DataNodes 中存储为块。在执行 MapReduce 作业期间，各个 Mapper 会处理块（输入拆分）。如果数据不在映射器执行作业的同一节点中，则需要通过网络将数据从 DataNode 复制到映射器 DataNode。

现在，如果一个 MapReduce 作业具有 100 个以上的 Mapper，并且每个 Mapper 尝试同时复制集群中其他 DataNode 的数据，则将导致严重的网络拥塞，这是整个系统的一个大性能问题。因此，数据接近计算是一种有效且具有成本效益的解决方案，在技术上被称为 Hadoop 中的数据本地性。它有助于提高系统的整体吞吐量。



数据局部性可以分为三种类型：

本地数据 - 这种类型的数据和映射器位于同一节点上。这是最接近的数据，也是最优选的方案。

本地机架 - 在这种情况下，映射器和数据位于同一机架上，但位于不同的数据节点上。

不同的机架 - 在这种情况下，映射器和数据位于不同的机架上。

131.【简答题】【Hadoop】什么是 Sequencefileinputformat？

Hadoop 使用一种特定的文件格式，即序列文件。序列文件将数据存储在序列化的键值对中。Sequencefileinputformat 是用于读取序列文件的输入格式。

132.【简答题】【Hadoop】Hadoop 的三种运行模式是什么？

Hadoop 的三种运行模式如下：

（1）独立或本地：这是默认模式，不需要任何配置。在这种模式下，Hadoop 的以下所有组件均使用本地文件系统，并在单个 JVM 上运行 -

名称节点

数据节点

资源管理器

节点管理器

（2）伪分布式：在这种模式下，所有主和从 Hadoop 服务都在单个节点上部署和执行。

（3）完全分布式：在这种模式下，Hadoop 主服务和从服务在单独的节点上部署和执行。

133.【简答题】【Hadoop】在 Hadoop 中解释 JobTracker

JobTracker 是 Hadoop 中的 JVM 流程，用于提交和跟踪 MapReduce 作业。

JobTracker 按顺序在 Hadoop 中执行以下活动 -

JobTracker 接收客户端应用程序提交给作业跟踪器的作业

JobTracker 通知 NameNode 确定数据节点

JobTracker 根据可用的插槽分配 TaskTracker 节点。

它在分配的 TaskTracker 节点上提交工作，

JobTracker 监视 TaskTracker 节点。

任务失败时，将通知 JobTracker 并决定如何重新分配任务。

134.【简答题】【Hadoop】Hadoop 中有哪些不同的配置文件？

Hadoop 中的不同配置文件是 -

core-site.xml - 此配置文件包含 Hadoop 核心配置设置，例如 I / O 设置，这对于 MapReduce 和 HDFS 非常常见。它使用主机名端口。

mapred-site.xml - 此配置文件通过设置 mapreduce.framework.name 为 MapReduce 指定框架名称

hdfs-site.xml - 此配置文件包含 HDFS 守护程序配置设置。它还在 HDFS 上指定默认阻止权限和复制检查。

yarn-site.xml - 此配置文件指定 ResourceManager 和 NodeManager 的配置设置。

135.【简答题】【Hadoop】Hadoop 2 和 Hadoop 3 有什么区别？

以下是 Hadoop 2 和 Hadoop 3 之间的区别 -

Criteria	Hadoop 2	Hadoop 3
Java Version	Minimum supported Java Version is Java 7	Minimum supported Java Version is Java 8
Fault Tolerance	Fault tolerance is handled by Replication that waste the space	Fault tolerance is handled by Erasure coding
Data Balancing	HDFS balancer is used for data balancing	Intra-datanode balancer is used for data balancing
Overhead in Storage Space	HDFS has 200% overhead in storage space	HDFS has 50% overhead in storage space
Default Ports	Some default ports are in ephemeral range, so fails to bind at start up	All the default ports are out of the ephemeral range, binds well at start up

136.【简答题】【Hadoop】如何在 Hadoop 中实现安全性？

Kerberos 用于在 Hadoop 中实现安全性。使用 Kerberos 时，共有 3 个步骤可以高层访问服务。每个步骤都涉及

与服务器的消息交换。

身份验证－第一步涉及到客户端对身份验证服务器的身份验证，然后向客户端提供带时间戳的 TGT（票证授予票证）。

授权－在此步骤中，客户端使用收到的 TGT 向 TGS（票证授予服务器）请求服务票证。

服务请求－这是在 Hadoop 中实现安全性的最后一步。然后，客户端使用服务票证向服务器进行身份验证。

137.【简答题】【Hadoop】什么是商品硬件？

商品硬件是一种可用性较低，质量较低的低成本系统。商品硬件由 RAM 组成，因为它执行许多需要 RAM 才能执行的服务。一个不需要高端硬件配置或超级计算机即可运行 Hadoop，它可以在任何商用硬件上运行。

138.【简答题】【Hadoop】Hadoop 中 jps 命令的用途是什么？

jps 命令用于检查 Hadoop 守护程序是否正常运行。此命令显示在计算机上运行的所有守护程序，即 Datanode，Namenode，NodeManager，ResourceManager 等。

139.【简答题】【Hadoop】定义 HDFS 和 YARN 的各个组件

HDFS 的两个主要组成部分是：

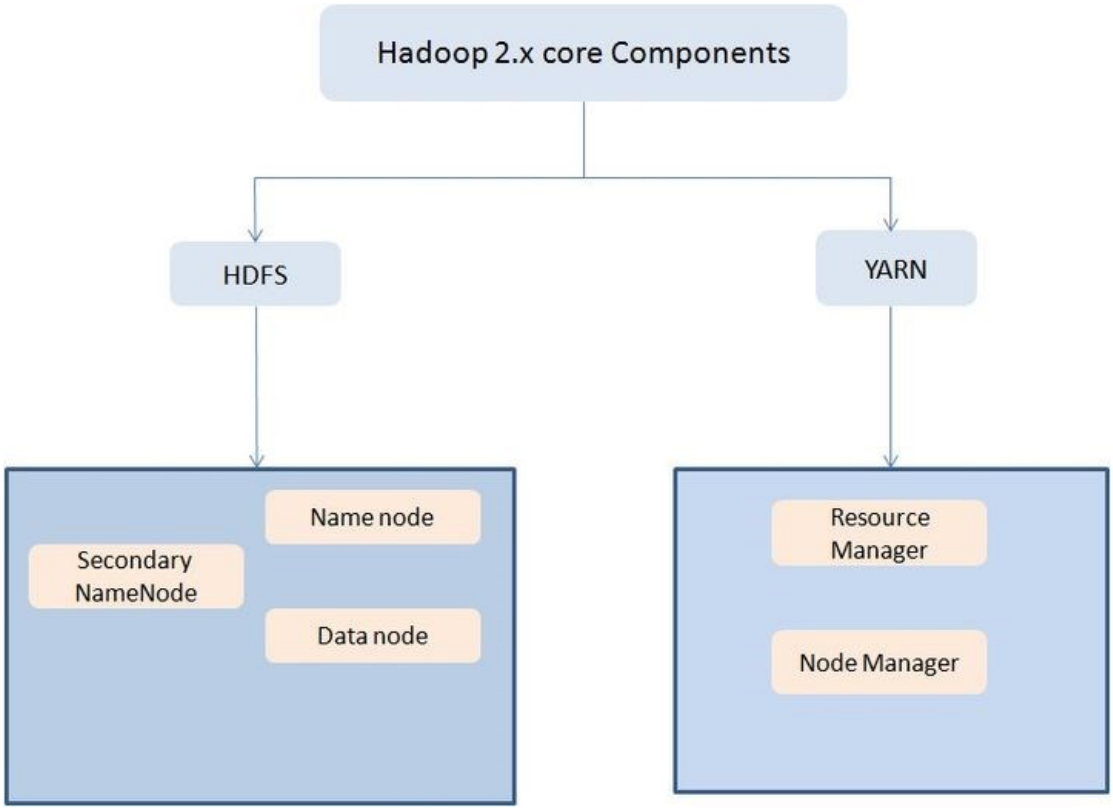
NameNode －这是主节点，用于处理 HDFS 中数据块的元数据信息

DataNode / Slave 节点－这是一个充当从节点存储数据，供 NameNode 处理和使用的节点

除了满足客户端请求之外，NameNode 还执行以下两个角色之一：

CheckpointNode －它在与 NameNode 不同的主机上运行

BackupNode-这是一个只读的 NameNode，其中包含文件系统元数据信息（不包括块位置）



YARN 的两个主要组成部分是：

ResourceManager －此组件接收处理请求，并根据处理需要相应地分配给相应的 NodeManager。

NodeManager －在每个数据节点上执行任务

140.【简答题】【Hadoop】为什么将 Hadoop 用于大数据分析？

由于数据分析已成为业务的关键参数之一，因此，企业正在处理大量的结构化，非结构化和半结构化数据。在 Hadoop 以其以下功能为主要角色的情况下，分析非结构化数据非常困难：

存储

处理中

数据采集

此外，Hadoop 是开源的，并在商品硬件上运行。因此，它是企业的成本效益解决方案。

141.【简答题】【Flink】Flink 任务，delay 极高，请问你有什么调优策略?

首先要确定问题产生的原因，找到最耗时的点，确定性能瓶颈点。比如任务频繁反压，找到反压点。主要通过:资源调优、作业参数调优。资源调优即是对作业中的 Operator 的并发数(parallelism)、CPU(core)、堆内存(heap_memory)等参数进行调优。作业参数调优包括:并行度的设置，State 的设置，checkpoint 的设置。

142.【简答题】【MapReduce】combine 出现在哪个过程，举例说明什么情况下可以使用 combiner，什么情况下不可以。

主要从以下角度进行阐述：

combiner 是发生在 map 的最后一个阶段，其原理也是一个小型的 reducer，主要作用是减少输出到 reduce 的个数，减少 reducer 的输入，提高 reducer 的执行效率。注意：mapper 的输出为 combiner 的输入，reducer 的输入为 combiner 的输出。

求平均数的时候就不需要用 combiner，因为不会减少 reduce 执行数量。在其他的时候，可以依据情况，使用 combiner，来减少 map 的输出数量，减少拷贝到 reduce 的文件，从而减轻 reduce 的压力，节省网络开销，提升执行效率。

143.【简答题】【MapReduce】简述 mapreduce 的运行原理

先将文件进行分割后，进行 map 操作，后面进行 shuffle 操作，分为 map 端 shuffle 和 reduce 端 shuffle，map 输出结果放在缓冲区，当缓存区到达一定阈值时，将其中数据 spill（也就是溢写）到磁盘，然后进行 partition, sort, combine 操作，这样多次 spill 后，磁盘上就会有多个文件，merge 操作将这些文件合并成一个文件，reduce 端 shuffle 从 map 节点拉取数据文件，如果在内存中放得下，就直接放在内存中，每个 map 对应一块数据，当内存占用量达到一定程度时，启动内存时 merge，把内存中的数据输出到磁盘的一个文件上。如果在内存中放不下的话，就直接写到磁盘上。一个 map 数据对应一个文件，当文件数量达到一定阀值时，开始启动磁盘文件 merge，把这些文件合并到一个文件中。最后，把内存中的文件和磁盘上的文件进行全局 merge，形成一个最终端文件，做为 reduce 的输入文件。当然 merge 过程中会进行 sort,combine 操作。

144.【简答题】【Mapreduce】请说下 MR 中 MapTask 的工作机制

简单概述:

inputFile 通过 split 被切割为多个 split 文件，通过 Record 按行读取内容给 map(自己写的处理逻辑的方法)，数据被 map 处理完之后交给 OutputCollect 收集器，对其结果 key 进行分区(默认使用的 hashPartitioner)，然后写入 buffer，每个 maptask 都有一个内存缓冲区(环形缓冲区)，存放着 map 的输出结果，当缓冲区快满的时候需要将缓冲区的数据以一个临时文件的方式溢写到磁盘，当整个 maptask 结束后再对磁盘中这个 maptask 产生的所有临时文件做合并，生成最终的正式输出文件，然后等待 reducetask 的拉取

详细步骤:

1)读取数据组件 InputFormat(默认 TextInputFormat)会通过 getSplits 方法对输入目录中的文件进行逻辑切片规划得到 block,有多少个 block 就对应启动多少个 MapTask.

2)将输入文件切分为 block 之后,由 RecordReader 对象(默认是 LineRecordReader)进行读取,以\n 作为分隔符,读取一行数据,返回<key，value\>.Key 表示每行首字符偏移值,Value 表示这一行文本内容

3)读取 block 返回<key,value\>,进入用户自己继承的 Mapper 类中，执行用户重写的 map 函数,RecordReader 读取

一行这里调用一次

4)Mapper 逻辑结束之后,将 Mapper 的每条结果通过 context.write 进行 collect 数据收集.在 collect 中,会先对其进行分区处理，默认使用 HashPartitioner

5)接下来,会将数据写入内存,内存中这片区域叫做环形缓冲区(默认 **100M),**缓冲区的作用是批量收集 **Mapper** 结果,减少磁盘 **IO** 的影响.我们的 **Key/Value** 对以及 **Partition** 的结果都会被写入缓冲区.当然,写入之前，**Key** 与 **Value** 值都会被序列化成字节数组

6)当环形缓冲区的数据达到溢写比列(默认 0.8)，也就是 80M 时，溢写线程启动,需要对这 **80MB** 空间内的 **Key** 做排序**(Sort)**.排序是 MapReduce 模型默认的行为,这里的排序也是对序列化的字节做的排序

7)合并溢写文件,每次溢写会在磁盘上生成一个临时文件(写之前判断是否有 Combiner),如果 Mapper 的输出结果真的很大,有多次这样的溢写发生,磁盘上相应的就会有多个临时文件存在.当整个数据处理结束之后开始对磁盘中的临时文件进行 Merge 合并,因为最终的文件只有一个,写入磁盘,并且为这个文件提供了一个索引文件,以记录每个 reduce 对应数据的偏移量

145.【简答题】【Mapreduce】请说下 MR 中 ReduceTask 的工作机制

简单描述:

Reduce 大致分为 copy、sort、reduce 三个阶段，重点在前两个阶段。copy 阶段包含一个 eventFetcher 来获取已完成的 map 列表，由 Fetcher 线程去 copy 数据，在此过程中会启动两个 merge 线程，分别为 in Memory Merger 和 on Disk Merger，分别将内存中的数据 merge 到磁盘和将磁盘中的数据进行 merge。待数据 copy 完成之后，copy 阶段就完成了，开始进行 sort 阶段，sort 阶段主要是执行 final Merge 操作，纯粹的 sort 阶段，完成之后就是 reduce 阶段，调用用户定义的 reduce 函数进行处理

详细步骤:

1)**Copy** 阶段:简单地拉取数据。Reduce 进程启动一些数据 copy 线程(Fetcher)，通过 HTTP 方式请求 maptask 获取属于自己的文件(maptask 的分区会标识每个 maptask 属于哪个 reducetask，默认 reducetask 的标识从 0 开始)。

2)**Merge** 阶段:这里的 merge 如 map 端的 merge 动作，只是数组中存放的是不同 map 端 copy 来的数值。Copy 过来的数据会先放入内存缓冲区中，这里的缓冲区大小要比 map 端的更为灵活。merge 有三种形式:内存到内存;内存到磁盘;磁盘到磁盘。默认情况下第一种形式不启用。当内存中的数据量到达一定阈值，就启动内存到磁盘的 merge。与 map 端类似，这也是溢写的过程，这个过程中如果你设置有 Combiner，也是会启用的，然后在磁盘中生成了众多的溢写文件。第二种 merge 方式一直在运行，直到没有 map 端的数据时才结束，然后启动第三种磁盘到磁盘的 merge 方式生成最终的文件。

3)合并排序:把分散的数据合并成一个大的数据后，还会再对合并后的数据排序。

4)对排序后的键值对调用 **reduce** 方法，键相等的键值对调用一次 reduce 方法，每次调用会产生零个或者多个键值对，最后把这些输出的键值对写入到 HDFS 文件中。

146.【简答题】【Mapreduce】请说下 MR 中 shuffle 阶段

shuffle 阶段分为四个步骤:依次为:分区，排序，规约，分组，其中前三个步骤在 map 阶段完成，最后一个步骤

在 reduce 阶段完成，shuffle 是 Mapreduce 的核心，它分布在 Mapreduce 的 map 阶段和 reduce 阶段。一般把从 Map 产生输出开始到 Reduce 取得数据作为输入之前的过程称作 shuffle。

1.**Collect** 阶段:将 MapTask 的结果输出到默认大小为 100M 的环形缓冲区，保存的是 key/value，Partition 分区信息等。

2.**Spill** 阶段:当内存中的数据量达到一定的阀值的时候，就会将数据写入本地磁盘，在将数据写入磁盘之前需要对数据进行一次排序的操作，如果配置了 combiner，还会将有相同分区号和 key 的数据进行排序。

3.**Merge** 阶段:把所有溢出的临时文件进行一次合并操作，以确保一个 MapTask 最终只产生一个中间数据文件

4.Copy 阶段:ReduceTask 启动 Fetcher 线程到已经完成 MapTask 的节点上复制一份属于自己的数据，这些数据默认会保存在内存的缓冲区中，当内存的缓冲区达到一定的阀值的时候，就会将数据写到磁盘之上

5.**Merge** 阶段:在 ReduceTask 远程复制数据的同时，会在后台开启两个线程对内存到本地的数据文件进行合并操作

6.**Sort** 阶段:在对数据进行合并的同时，会进行排序操作，由于 MapTask 阶段已经对数据进行了局部的排序，ReduceTask 只需保证 Copy 的数据的最终整体有效性即可。

147.【简答题】【Mapreduce】在写 MR 时，什么情况下可以使用规约

规约(combiner)是不能够影响任务的运行结果的，局部汇总，适用于求和类，不适用于求平均值，如果 reduce 的输入参数类型和输出参数的类型是一样的，则规约的类可以使用 reduce 类，只需要在驱动类中指明规约的类即可

148.【简答题】【redis、hbase、hive、RDBS】Redis, 传统数据库,hbase,hive 每个之间的区别？

1）Redis 是基于内存的数据库，注重实用内存的计算，

2）hbase 是列式数据库，无法创建主键，地从是基于 HDFS 的，每一行可以保存很多的列，

3）hive 是数据的仓库，是为了减轻 mapreduce 而设计的，不是数据库，是用来与 hadoop 做交互的。

149.【简答题】【zookeeper】你对 zookeeper 的理解？

1）随着大数据的快速发展，多机器的协调工作，避免主要机器单点故障的问题，于是就引入管理机器的一个软件，他就是 zookeeper 来协助机器正常的运行。

2）Zookeeper 有两个角色分别是 leader 与 follower ，其中 leader 是主节点，其他的是副节点，在安装配置上一定要注意配置奇数个的机器上，便于 zookeeper 快速切换选举其他的机器。

3）在其他的软件执行任务时在 zookeeper 注册时会在 zookeeper 下生成相对应的目录，以便 zookeeper 去管理机器。

150.【简答题】【zookeeper】ZooKeeper 提供了什么？

1、文件系统

2、通知机制

151.【简答题】【Zookeeper】 ZAB 协议？

ZAB 协议是为分布式协调服务 Zookeeper 专门设计的一种支持崩溃恢复的原子广播协议。

ZAB 协议包括两种基本的模式：崩溃恢复和消息广播。

当整个 zookeeper 集群刚刚启动或者 Leader 服务器宕机、重启或者网络故障导致不存在过半的服务器与 Leader 服务器保持正常通信时，所有进程（服务器）进入崩溃恢复模式，首先选举产生新的 Leader 服务器，然后集群中 Follower 服务器开始与新的 Leader 服务器进行数据同步，当集群中超过半数机器与该 Leader 服务器完成数据同步之后，退出恢复模式进入消息广播模式，Leader 服务器开始接收客户端的事务请求生成事物提案来进行事务请求处理。

152.【简答题】【zookeeper】数据同步

整个集群完成 Leader 选举之后，Learner（Follower 和 Observer 的统称）回向 Leader 服务器进行注册。当 Learner 服务器想 Leader 服务器完成注册后，进入数据同步环节。

153.【简答题】【zookeeper】集群最少要几台机器，集群规则是怎样的?

集群规则为 2N+1 台，N>0，即 3 台。

154.【简答题】【zookeeper】说几个 zookeeper 常用的命令

常用命令：ls get set create delete 等。

155.【简答题】【Yarn】yarn 的任务提交流程是怎样的

当 jobclient 向 YARN 提交一个应用程序后，YARN 将分两个阶段运行这个应用程序:一是启动 ApplicationMaster;第二个阶段是由 ApplicationMaster 创建应用程序，为它申请资源，监控运行直到结束。

具体步骤如下:

1)用户向 YARN 提交一个应用程序，并指定 ApplicationMaster 程序、启动 ApplicationMaster 的命令、用户程序。

2)RM 为这个应用程序分配第一个 Container，并与之对应的 NM 通讯，要求它在这个 Container 中启动应用程序 ApplicationMaster。

3)ApplicationMaster 向 RM 注册，然后拆分为内部各个子任务，为各个内部任务申请资源，并监控这些任务的运行，直到结束。

4)AM 采用轮询的方式向 RM 申请和领取资源。

5)RM 为 AM 分配资源，以 Container 形式返回

6)AM 申请到资源后，便与之对应的 NM 通讯，要求 NM 启动任务。

7)NodeManager 为任务设置好运行环境，将任务启动命令写到一个脚本中，并通过运行这个脚本启动任务

8)各个任务向 AM 汇报自己的状态和进度，以便当任务失败时可以重启任务。

9)应用程序完成后，ApplicationMaster 向 ResourceManager 注销并关闭自己

156.【多选题】【HBase】 HBase 官方版本可以安装在什么操作系统上？（ ）

A、CentOS

B、Ubuntu

C、RedHat

D、Windows

答案：A、B、C。

可以安装到 Linux 操作系统上，不能安装到 Windows 操作系统上。

157.【单选题】【HBase】下列关于 HBase 平台部署描述错误的是？（ ）

A、HBASE_MANAGES_ZK=true,表示由 hbase 自己管理 zookeeper，不需要单独的部署 zookeeper

B、hbase.rootdir 用于指定 HBase 数据的存储位置

C、hbase.zookeeper.property 指定 zookeeper 数据存储目录，默认路径是/tmp，如果不配置，重启之后数据不会被清空

D、hbase.zookeeper.quorum 指定使用 zookeeper 的主机地址

答案： C。

解析：hbase.zookeeper.property 指定 zookeeper 数据存储目录，默认路径是/tmp，如果不配置，重启之后数据会被清空。

158.【单选题】【HBase】 HBase 依靠（）存储底层数据（ ）

A、HDFS

B、Hadoop

C、Memory

D、MapReduce

答案： A。

HBase 依赖于 Hadoop 中的 HDFS。

159.【单选题】【HBase】 HBase 依赖（ ）提供强大的计算能力。

A、Zookeeper

B、Chubby

C、RPC

D、MapReduce

答案： D。

解析：HBase 依赖 MapReduce 提供计算能力。

160.【单选题】【HBase】 关于 HBase 服务启动正确的是（ ）

A、bin/start-hbase.sh

B、sbin/hbase-daemon.sh start RegionServer

C、sbin/start-all.sh

D、sbin/start-hbase.sh

答案：A。

解析：Hbase 没有 sbin 目录。

161.【多选题】【HBase】 在$HBASE_HOME/bin 下执行命令：start-hbase.sh 启动之后，HBase 的守护进程分别是（ ）

A、HQuorumPeer

B、HRegionServer

C、NameNode

D、HMaster

答案：A、B、D。

解析：NameNode 是 Hadoop 的守候进程。

162.【单选题】【HBase】 关于 HBase 与 MapReduce 集成说法错误的是（ ）

A、一个 MapReduce 作业中的源和目标都可以是 HBase 表，但不能够在一个作业中同时使用 HBase 作为输入和输出

B、使用 MapReduce 作业作为过程的一部分，该作业可以使用 HBase 进行读取或写入

C、用户可以在存储组合结果的不同表中查询数据。用户从哪里读数据和向哪里写数据是不受限制的

D、通过 MapReduce 的 API 实现可以以任何形式访问 HBase 的代码

答案：A。

解析：NameNode 是 Hadoop 的守候进程一个 MapReduce 作业中的源和目标都可以是 HBase 表，但也有可能在一个作业中同时使用 HBase 作为输入和输出。

163.【简答题】【Hbase】向 Hbase 中写入数据时，直接将时间戳作为行健，在写入单个 region 时候会发生热点问题，为什么呢？

region 中的 rowkey 是有序存储，若时间比较集中。就会存储到一个 region 中，这样一个 region 的数据变多，其它的 region 数据很少，加载数据就会很慢，直到 region 分裂，此问题才会得到缓解

164.【简答题】【HBase】HBase 是怎么读取数据的？

1)Client 先访问 zookeeper，从 meta 表读取 region 的位置，然后读取 meta 表中的数据。meta 中又存储了用户表的 region 信息;

2)根据 namespace、表名和 rowkey 在 meta 表中找到对应的 region 信息;

3)找到这个 region 对应的 regionserver;

4)查找对应的 region;

5)先从 MemStore 找数据，如果没有，再到 Block Cache 里面读;

6)Block Cache 还没有，再到 Store File 上读(为了读取的效率);

7)如果是从 Store File 里面读取的数据，不是直接返回给客户端，而是先写入 Block Cache，再返回给客户端。

165.【简答题】【HBase】H Base 的存储结构

Hbase 中的每张表都通过行键(rowkey)按照一定的范围被分割成多个子表(HRegion)，默认一个 HRegion 超过 256M 就要被分割成两个，由 HRegionServer 管理，管理哪些 HRegion 由 Hmaster 分配。HRegion 存取一个子表时，会创建一个 HRegion 对象，然后对表的每个列族(ColumnFamily)创建一个 store 实例，每个 store 都会有 0 个或多个 StoreFile 与之对应，每个 StoreFile 都会对应一个 HFile，HFile 就是实际的存储文件，一个 HRegion 还拥有一个

MemStore 实例。

166.【简答题】【Hbase】请描述如何解决 HBase 中 region 太小和 region 太大带来的冲突？

Region 过大会发生多次 compaction，将数据读一遍并重写一遍到 hdfs 上，占用 io，region 过小会造成多次 split，region 会下线，影响访问服务，最佳的解决方法是调整 hbase.hregion. max.filesize 为 256m

167.【简答题】【HBase】HDFS 和 HBase 各自使用场景

首先一点需要明白:Hbase 是基于 HDFS 来存储的。

HDFS:

一次性写入，多次读取。

保证数据的一致性。

主要是可以部署在许多廉价机器中，通过多副本提高可靠性，提供了容错和恢复机制。

HBase:

瞬间写入量很大，数据库不好支撑或需要很高成本支撑的场景。

数据需要长久保存，且量会持久增长到比较大的场景。

HBase 不适用与有 join，多级索引，表关系复杂的数据模型。

大数据量(100sTB 级数据)且有快速随机访问的需求。如:淘宝的交易历史记录。数据量巨大无容置疑，面向普通用户的请求必然要即时响应

业务场景简单，不需要关系数据库中很多特性(例如交叉列、交叉表，事务，连接等等)。

168.【简答题】【HBase】HBase 的 rowkey 设计原则

长度原则:100 字节以内，8 的倍数最好，可能的情况下越短越好。因为 HFile 是按照 keyvalue 存储的，过长的 rowkey 会影响存储效率;其次，过长的 rowkey 在 memstore 中较大，影响缓冲效果，降低检索效率。最后，操作系统大多为 64 位，8 的倍数，充分利用操作系统的最佳性能。

散列原则:高位散列，低位时间字段。避免热点问题。

唯一原则:分利用这个排序的特点，将经常读取的数据存储到一块，将最近可能会被访问的数据放到一块。

169.【简答题】【HBase】Hbase 是怎么写数据的?

Client 写入->存入 MemStore，一直到 MemStore 满->Flush 成一个 StoreFile，直至增长到一定阈值->触发 Compact 合并操作->多个 StoreFile 合并成一个 StoreFile，同时进行版本合并和数据删除->当 StoreFilesCompact 后，逐步形成越来越大的 StoreFile->单个 StoreFile 大小超过一定阈值后(默认 10G)，触发 Split 操作，把当前 RegionSplit

成 2 个 Region，Region 会下线，新 Split 出的 2 个孩子 Region 会被 HMaster 分配到相应的 HRegionServer 上，使得原先 1 个 Region 的压力得以分流到 2 个 Region 上。

由此过程可知，HBase 只是增加数据，没有更新和删除操作，用户的更新和删除都是逻辑层面的，在物理层面，更新只是追加操作，删除只是标记操作。

用户写操作只需要进入到内存即可立即返回，从而保证 I/O 高性能。

170.【简答题】【Hbase】rowkey 设计原则？

rowkey 长度原则

rowkey 散列原则

rowkey 唯一原则

171.【简答题】【Hbase】Rowkey 如何设计？

生成随机数、hash、散列值

字符串反转

172.【简答题】【Hbase】HBase 的特点是什么？

大：一个表可以有数十亿行，上百万列

无模式：每行都有一个可排序的主键和任意多的列，列可以根据需要动态的增加，同一张表中不同的行可以有截然不同的列

面向列：面向列（族）的存储和权限控制，列（族）独立检索

稀疏：空（null）列并不占用存储空间，表可以设计的非常稀疏

数据多版本：每个单元中的数据可以有多个版本，默认情况下版本号自动分配，是单元格插入时的时间戳

数据类型单一：Hbase 中的数据都是字符串，没有类型

173.【简答题】【Hbase】 HBase 和 Hive 的区别？

Hive 和 Hbase 是两种基于 Hadoop 的不同技术--Hive 是一种类 SQL 的引擎，并且运行 MapReduce 任务，Hbase 是一种在 Hadoop 之上的 NoSQL 的 Key/vale 数据库。当然，这两种工具是可以同时使用的。就像用 Google 来搜索，用 FaceBook 进行社交一样，Hive 可以用来进行统计查询，HBase 可以用来进行实时查询，数据也可以从 Hive 写到 Hbase，设置再从 Hbase 写回 Hive

Apache Hive 是一个构建在 Hadoop 基础设施之上的数据仓库。通过 Hive 可以使用 HQL 语言查询存放在 HDFS 上的数据。HQL 是一种类 SQL 语言，这种语言最终被转化为 Map/Reduce. 虽然 Hive 提供了 SQL 查询功能，但是 Hive 不能够进行交互查询--因为它只能够在 Haoop 上批量的执行 Hadoop

Apache HBase 是一种 Key/Value 系统，它运行在 HDFS 之上。和 Hive 不一样，Hbase 的能够在它的数据库上实

时运行，而不是运行 MapReduce 任务。Hbase 被分区为表格，表格又被进一步分割为列簇。列簇必须使用 schema 定义，列簇将某一类型列集合起来（列不要求 schema 定义）。例如，“message”列簇可能包含：“to”，”from” “date”，“subject”，和”body”．每一个 key/value 对在 Hbase 中被定义为一个 cell，每一个 key 由 row-key，列簇、列和时间戳。在 Hbase 中，行是 key/value 映射的集合，这个映射通过 row-key 来唯一标识。Hbase 利用 Hadoop 的基础设施，可以利用通用的设备进行水平的扩展。

174.【简答题】【Hbase】HBase 适用于怎样的情景？

半结构化或非结构化数据

记录非常稀疏

多版本数据

超大数据量

175.【简答题】【HBase】请详细描述 HBase 中一个 cell 的结构？

HBase 中通过 row 和 columns 确定的为一个存贮单元称为 cell。

Cell：由{row key, column(= +), version}唯一确定单元。cell 中的数据是没有类型的，全部是字节码形式存贮。

176.【简答题】【HBase】.HBase 内部机制是什么？

Hbase 是一个能适应联机业务的数据库系统

物理存储：hbase 的持久化数据是将数据存储在 HDFS 上。

存储管理：一个表是划分为很多 region 的，这些 region 分布式地存放在很多 regionserver 上 Region 内部还可以划分为 store， store 内部有 memstore 和 storefile。

版本管理：hbase 中的数据更新本质上是不断追加新的版本，通过 compact 操作来做版本间的文件合并 Region 的 split。

集群管理：ZooKeeper + HMaster + HRegionServer。

177.【简答题】【HBase】怎样将 mysql 的数据导入到 hbase 中？

不能使用 sqoop，速度太慢了，提示如下：

A、一种可以加快批量写入速度的方法是通过预先创建一些空的 regions，这样当数据写入 HBase 时，会按照 region 分区情况，在集群内做数据的负载均衡。

B、hbase 里面有这样一个 hfileoutputformat 类，他的实现可以将数据转换成 hfile 格式，通过 new 一个这个类，进行相关配置,这样会在 hdfs 下面产生一个文件，这个时候利用 hbase 提供的 jruby 的 loadtable.rb 脚本就可以进行批量导入。

178.【简答题】【Flink】Flink 中的时间种类有哪些?各自介绍一下?

Flink 中的时间与现实世界中的时间是不一致的，在 flink 中被划分为事件时间，摄入时间，处理时间三种。如果以 EventTime 为基准来定义时间窗口将形成 EventTimeWindow,要求消息本身就应该携带 EventTime 如果以 IngesingtTime 为基准来定义时间窗口将形成

IngestingTimeWindow,以 source 的 systemTime 为准。如果以 ProcessingTime 基准来定义时间窗口将形成 ProcessingTimeWindow，以 operator 的 systemTime 为准。

179.【简答题】【Flink】Flink 的 table 和 SQL 熟悉吗?TableAPI 和 SQL 中 TableEnvironment 这个类有什么作用

TableEnvironment 是 TableAPI 和 SQL 集成的核心概念。它负责:

A)在内部 catalog 中注册表

B)注册外部 catalog

C)执行 SQL 查询

D)注册用户定义(标量，表或聚合)函数

E)将 DataStream 或 DataSet 转换为表

F)持有对 ExecutionEnvironment 或 StreamExecutionEnvironment 的引用

180.【简答题】【Flink】Flink 如何实现 SQL 解析的呢?

StreamSQLAPI 的执行原理如下:

1、用户使用对外提供 StreamSQL 的语法开发业务应用;

2、用 calcite 对 StreamSQL 进行语法检验，语法检验通过后，转换成 calcite 的逻辑树节点;最终形成 calcite 的逻辑计划;

3、采用 Flink 自定义的优化规则和 calcite 火山模型、启发式模型共同对逻辑树进行优化，生成最优的 Flink 物理计划;

4、对物理计划采用 janinocodegen 生成代码，生成用低阶 APIDataStream 描述的流应用，提交到 Flink。

181.【单选题】【MapReduce】关于 MapReduce 过程各个角色的作用说法不正确的是（ ）

A、JobTracker 负责接收用户提交的作业，负责启动、跟踪任务执行

B、TaskTracker 负责执行任务，负责启动、跟踪任务执行、访问任务状态和日志

C、JobClient 负责提交作业

D、客户端提交一个 MR 的 jar 包给 JobTracker，再由 JobTracker 分给 TaskTracker

答案：C。

解析：TaskTracker 负责执行任务、JobClient 负责提交作业的，负责启动、跟踪任务执行、访问任务状态和日志。

182.【单选题】【MapReduce】关于 MapReduce 执行过程说法错误的是（　　）

A、MapReduce 执行过程分为 Mapper 过程与 Reducer 过程

B、合并、排序、Shuffle 是在 Mapper 输出之后，Reducer 输出之前完成的

C、Mapper 过程与 Reducer 过程之间，会有 shuffle 过程

D、正常执行 MapReduce 任务时，Map 函数和 reduce 函数都会被调用执行

答案：D。

解析：正常执行 MapReduce 任务时，Map 函数会调用，而 reduce 函数不一定执行。

183.【单选题】【MapReduce】下列描述中不符合 Map/Reduce 的是(　　)

A、Map 是将数据映射成 Key/Value 再交给 Reduce

B、Reduce 先运行，然后运行 Map

C、Map/Reduce 是函数式的设计思想

D、Map 结束后，Partitioner 会将相同 Key 分到同一个组交给 Reduce 进程

答案：B。

解析：Map 结束后，才会运行 Reduce。

184.【单选题】【MapReduce】有关 MapReduce 的输入输出，说法错误的是(　　)

A、链接多个 MapReduce 作业时，序列文件是首选格式

B、把输入数据划分为分片，分片数目和大小任意定义

C、想完全禁止输出，可以使用 Null Output Format

D、每个 reduce 需将它的输出写入自己的文件中，输出无需分片

答案：B。

解析：分片数目和大小不能任意定义。

185.【单选题】【MapReduce】有关 MapReduce 的输入输出，说法错误的是(　　)

A、 链接多个 MapReduce 作业时，序列文件是首选格式

B、FileInputFormat 中实现的 getSplits()可以把输入数据划分为分片，分片数目和大小任意定义

C、想完全禁止输出，可以使用 NullOutputFormat

D、每个 reduce 需将它的输出写入自己的文件中，输出无需分

答案：B。

解析：分片数目在 numSplits 中限定，分片大小必须大于 mapreD、min.size 个字节，但小于文件系统的块。

186.【单选题】【MapReduce】下面关于 MapReduce 的描述中正确的是(　　)

A、MapReduce 程序必须包含 Mapper 和 Reducer

B、MapReduce 程序的 MapTask 可以任意指定

C、MapReduce 程序的 ReduceTask 可以任意指定

D、MapReduce 程序的默认数据读取组件是 TextInputFormatB

答案：C。

解析：分片数目在 numSplits 中限定，分片大小必须大于 mapreD、min.size 个字节，但小于文件系统的块。

187.【单选题】【MapReduce】MapReduce 编程模型中以下组件哪个是最后执行的？(　　)

A、Mapper

B、Partitioner

C、Reducer

D、RecordReader

答案：C。

解析：以上这四个 MapReduce 编程模型中的执行顺序是：recordReader --> mapper --> partitioner --> reducer。

188.【单选题】【MapReduce】关于 MapReduce 处理类说法不正确的是(　　)

A、FileInputFormat 保存作为 job 输入的所有文件，并实现了对输入文件计算 splits 的方法

B、TextOutputformat 默认的输出格式，key 和 value 中间值用空格隔开的

C、InputSplit 原始数据被分割成若干 split，每个 split 作为一个 map 任务的输入

D、CombineFileInputFormat 相对于大量的小文件来说，更合适处理少量的大文件

答案：B。

解析：TextOutputformat 默认的输出格式，key 和 value 中间值不一定用空格隔开的。

189.【简答题】【MapReduce】 简述 MapReduce 中 Reducer 任务的执行过程？

主要从以下三方面进行阐述：

（1）第一阶段是 Reducer 任务会主动从 Mapper 任务复制其输出的键值对。Mapper 任务可能会有很多，因此 Reducer 会复制多个 Mapper 的输出。

（2）第二阶段是把复制到 Reducer 本地数据，全部进行合并，即把分散的数据合并成一个大的数据。再对合并后的数据排序。

（3）第三阶段是对排序后的键值对调用 reduce 方法,键相等的键值对调用一次 reduce 方法，每次调用会产生零个或者多个键值对。最后把这些输出的键值对写入到 HDFS 文件中。

190.【简答题】【MapReduce】Hadoop MapReduce 如何工作？

MapReduce 操作分为两个阶段。

映射阶段 - 在此阶段，输入数据由映射任务拆分。地图任务并行运行。这些拆分数据用于分析目的。

减少阶段-在此阶段，从整个集合中汇总相似的拆分数据并显示结果。

191.【简答题】【MapReduce】什么是 MapReduce？您运行 MapReduce 程序使用的语法是什么？

MapReduce 是 Hadoop 中的一种编程模型，用于在计算机集群（通常称为 HDFS）上处理大型数据集。它是一个并行编程模型。

运行 MapReduce 程序的语法为 - hadoop_jar_file.jar / input_path / output_path 。

192.【简答题】【MapReduce】Mapper 的基本参数是什么？

映射器的基本参数是

长写和文本

文字和可写

193.【简答题】【MapReduce】什么是 MapReduce 框架中的分布式缓存

分布式缓存是 Hadoop MapReduce 框架的一项功能，用于缓存应用程序的文件。Hadoop 框架使缓存文件可用于数据节点上运行的每个映射/减少任务。因此，数据文件可以在指定作业中作为本地文件访问缓存文件。

194.【简答题】【MapReduce】“ MapReduce”程序中的配置参数是什么？

“ MapReduce”框架中的主要配置参数为：

作业在分布式文件系统中的输入位置

作业在分布式文件系统中的输出位置

数据输入格式

数据输出格式

包含 map 函数的类

包含 reduce 函数的类

JAR 文件，其中包含映射器，reducer 和驱动程序类

195.【多选题】【Spark】以下哪些是 Spark-SQL 支持的数据格式和编程语言？( )

A、Scala

B、JSON

C、HiveQL

D、HDFS

答案：A、B、C、D。

196.【多选题】【Spark】RDD 的容错周期比较长，对于 Task 执行失败的恢复成本就比较高。那么 Spark 采取什么方式将数据冗余保存下来的呢？( )

A、检查点

B、资源共享

C、溢出

D、缓存

答案：A、D。

197.【单选题】【Spark】例如 Person 是一个 Java 对象，以下哪种结构信息或模式代表 DataFrame？( )

A、[Person]

B、[[Name][Age][Height]]

C、[[Name][Age Height]]

D、[[Person][Name Age Height]]

答案：B。

解析：DataFrame 是一种以 RDD 为基础的分布式数据集，也就是分布式的 Row 对象的集合（每个 Row 对象代表一行记录），提供了详细的结构信息。

198.【单选题】【Spark】Spark Streaming 的输入数据按照____分成一段一段的 DStream，每一段数据转换为 Spark 中的 RDD，并且对 DStream 的操作都最终转变为对相应的 RDD 的操作。(　　)

A、 数据量

B、 时间片

C、 复杂度

D、 来源类型

答案：B。

199.【单选题】【Spark】当参数环境：6 个 executor，kafka topic 有 3 个 partition，spark.streaming.concurrent.job=1 时，则：只有多少个 exevutor 有 task 在跑？(　　)

A、3

B、2

C、1

D、5

答案：A。

解析：一个 executor 处理一个分区数据

200.【单选题】【Spark】Spark 支持从兼容 HDFS API 的文件系统中读取数据，创建数据流。以下操作哪一个是创建文件数据流？

A、ssc.textFileStream()

B、ssc.textFile()

C、ssc.textFileStream()

D、ss.textFile()

答案：A。

解析：ssc.textFileStream()是从兼容 HDFS API 的文件系统中读取数据，创建数据流。

201.【单选题】【Spark】Kafka 是非常流行的日志采集系统，可以作为 DStream 的高级数据源。下面哪个操作可以读取 Kafka 数据流？

A、KafkaUtils.createStream

B、ssc.KafkaStream

C、KafkaUtils.buildStream

D、ssc.KafkaTextStream

答案：A。

解析：KafkaUtils.createStream 可以读取 Kafka 数据流

202.【单选题】【Spark】Spark Streaming 提供了窗口的计算，它允许你通过滑动窗口对数据进行转换,基于滑动窗口对（K，V）键值对类型的 DStream 中的值按 K 使用聚合函数 func 进行聚合操作是下列选项中的哪一个？(　　)

A、countByWindow

B、reduceByWindow

C、reduceByKeyAndWindow

D、countByValueAndWindow

答案：C。

解析：reduceByKeyAndWindow 是基于滑动窗口对（K，V）键值对类型的 DStream 中的值按 K 使用聚合函数 func 进行聚合操作，得到一个新的 DStream。

203.【简答题】【Spark】简单描述 Spark 的几种部署模式？

答案：

1) 本地模式

2) standalone 模式:分布式部署集群，自带完整的服务，资源管理和任务监控是 spark 自己监控

3) spark on yarn 模式:分布式部署集群，资源和任务监控交给 yarn 管理,目前只支持粗粒度资源分配方式

4) mesos 模式: 运行在 mesos 资源管理器框架之上，由 mesos 负责资源管理,分为粗粒度模式和细粒度模式

204.【简答题】【Spark】Spark 的数据本地性有哪几种？

Spark 中的数据本地性有三种：

1）PROCESS_LOCAL 是指读取缓存在本地节点的数据

2）NODE_LOCAL 是指读取本地节点硬盘数据

3）ANY 是指读取非本地节点数据

通常读取数据 PROCESS_LOCAL>NODE_LOCAL>ANY，尽量使数据以 PROCESS_LOCAL 或 NODE_LOCAL 方式读取。其中 PROCESS_LOCAL 还和 cache 有关，如果 RDD 经常用的话将该 RDDcache 到内存中，注意，由于 cache 是 lazy 的，所以必须通过一个 action 的触发，才能真正的将该 RDDcache 到内存中。

205.【简答题】【Spark】SparkonYarn 模式有哪些优点？

1）与其他计算框架共享集群资源（Spark 框架与 MapReduce 框架同时运行，如果不用 Yarn 进行资源分配，MapReduce 分到的内存资源会很少，效率低下）；资源按需分配，进而提高集群资源利用等

2）相较于 Spark 自带的 Standalone 模式，Yarn 的资源分配更加细致

3）Application 部署简化，例如 Spark，Storm 等多种框架的应用由客户端提交后，由 Yarn 负责资源的管理和调度，利用 Container 作为资源隔离的单位，以它为单位去使用内存,cpu 等

4）Yarn 通过队列的方式，管理同时运行在 Yarn 集群中的多个服务，可根据不同类型的应用程序负载情况，调整对应的资源使用量，实现资源弹性管理

206.【简答题】【Spark】谈谈你对 container 的理解？

1）Container 作为资源分配和调度的基本单位，其中封装了的资源如内存，CPU，磁盘，网络带宽等。目前 yarn 仅仅封装内存和 CPU

2）Container 由 ApplicationMaster 向 ResourceManager 申请的，由 ResouceManager 中的资源调度器异步分配给 ApplicationMaster

3）Container 的运行是由 ApplicationMaster 向资源所在的 NodeManager 发起的，Container 运行时需提供内部执行的任务命令

207.【简答题】【Spark】介绍 parition 和 block 有什么关联关系？

1）hdfs 中的 block 是分布式存储的最小单元，等分，可设置冗余，这样设计有一部分磁盘空间的浪费，但是整齐的 block 大小，便于快速找到、读取对应的内容

2）Spark 中的 partion 是弹性分布式数据集 RDD 的最小单元，RDD 是由分布在各个节点上的 partion 组成的。partion 是指的 spark 在计算过程中，生成的数据在计算空间内最小单元，同一份数据（RDD）的 partion 大小不一，数量不定，是根据 application 里的算子和最初读入的数据分块数量决定

3）block 位于存储空间、partion 位于计算空间，block 的大小是固定的、partion 大小是不固定的，是从 2 个不

同的角度去看数据

208.【简答题】【Spark】Spark 应用程序的执行过程是什么？

1）构建 SparkApplication 的运行环境（启动 SparkContext），SparkContext 向资源管理器（可以是 Standalone、Mesos 或 YARN）注册并申请运行 Executor 资源

2）资源管理器分配 Executor 资源并启动 StandaloneExecutorBackend，Executor 运行情况将随着心跳发送到资源管理器上

3）SparkContext 构建成 DAG 图，将 DAG 图分解成 Stage，并把 Taskset 发送给 TaskScheduler。Executor 向 SparkContext 申请 Task，TaskScheduler 将 Task 发放给 Executor 运行同时 SparkContext 将应用程序代码发放给 Executor

4）Task 在 Executor 上运行，运行完毕释放所有资源。

209.【简答题】【Spark】简要描述 Spark 写数据的流程？

1）RDD 调用 compute 方法，进行指定分区的写入

2）CacheManager 中调用 BlockManager 判断数据是否已经写入，如果未写，则写入

3）BlockManager 中数据与其他节点同步

4）BlockManager 根据存储级别写入指定的存储层

5）BlockManager 向主节点汇报存储状态中

210.【简答题】【Spark】Spark 有哪两种算子？

Transformation（转化）算子和 Action（执行）算子。

211.【简答题】【Spark】Spark 有哪些聚合类的算子,我们应该尽量避免什么类型的算子？

在我们的开发过程中，能避免则尽可能避免使用 reduceByKey、join、distinct、repartition 等会进行 shuffle 的算子，尽量使用 map 类的非 shuffle 算子。这样的话，没有 shuffle 操作或者仅有较少 shuffle 操作的 Spark 作业，可以大大减少性能开销。

212.【简答题】【Spark】Spark 工作机制?

用户在 client 端提交作业后，由 Driver 运行 main 方法并创建 spark context 上下文。

执行 add 算子，形成 dag 图，输入 dag scheduler，按照 add 之间的依赖关系划分 stage 输入 task scheduler。

Task scheduler 会将 stage 划分为 task set 分发到各个节点的 executor 中执行。

213.【简答题】【Spark】如何从 Kafka 中获取数据？

1）基于 Receiver 的方式

这种方式使用 Receiver 来获取数据。Receiver 是使用 Kafka 的高层次 ConsumerAPI 来实现的。receiver 从 Kafka 中获取的数据都是存储在 SparkExecutor 的内存中的，然后 SparkStreaming 启动的 job 会去处理那些数据。

2）基于 Direct 的方式

这种新的不基于 Receiver 的直接方式，是在 Spark1.3 中引入的，从而能够确保更加健壮的机制。替代掉使用 Receiver 来接收数据后，这种方式会周期性地查询 Kafka，来获得每个 topic+partition 的最新的 offset，从而定义每个 batch 的 offset 的范围。当处理数据的 job 启动时，就会使用 Kafka 的简单 consumerapi 来获取 Kafka 指定 offset 范围的数据。

214.【简答题】【Spark】Spark 如何处理不能被序列化的对象？

将不能序列化的内容封装成 object。

215.【简答题】【Spark】collect 功能是什么，其底层是怎么实现的？

driver 通过 collect 把集群中各个节点的内容收集过来汇总成结果，collect 返回结果是 Array 类型的，collect 把各个节点上的数据抓过来，抓过来数据是 Array 型，collect 对 Array 抓过来的结果进行合并，合并后 Array 中只有一个元素，是 tuple 类型（KV 类型的）的。

216.【简答题】【Spark】map 与flatMap 的区别？

map：对 RDD 每个元素转换，文件中的每一行数据返回一个数组对象

flatMap：对 RDD 每个元素转换，然后再扁平化将所有的对象合并为一个对象，文件中的所有行数据仅返回一个数组对象，会抛弃值为 null 的值

217.【简答题】【Spark】driver 的功能是什么？

一个 Spark 作业运行时包括一个 Driver 进程，也是作业的主进程，具有 main 函数，并且有 SparkContext 的实例，是程序的入口点。

功能：负责向集群申请资源，向 master 注册信息，负责了作业的调度，负责作业的解析、生成 Stage 并调度 Task 到 Executor 上，包括 DAGScheduler，TaskScheduler

218.【简答题】【Spark】Spark 中 Worker 的主要工作是什么？

主要功能：

管理当前节点内存，CPU 的使用状况，接收 master 分配过来的资源指令，通过 ExecutorRunner 启动程序分配任务，worker 就类似于包工头，管理分配新进程，做计算的服务，相当于 process 服务。

需要注意的点

1）worker 会不会汇报当前信息给 master，worker 心跳给 master 主要只有 workid，它不会发送资源信息以心跳的方式给 mater，master 分配的时候就知道 work，只有出现故障的时候才会发送资源

2）worker 不会运行代码，具体运行的是 Executor 是可以运行具体 appliaction 写的业务逻辑代码，操作代码的节点，它不会运行程序的代码的

219.【简答题】【Spark】MapReduce 和 Spark 的都是并行计算，那么他们有什么相同和区别？

两者都是用 mr 模型来进行并行计算：

1）hadoop 的一个作业称为 job，job 里面分为 maptask 和 reducetask，每个 task 都是在自己的进程中运行的，当 task 结束时，进程也会结束

2）spark 用户提交的任务成为 application，一个 application 对应一个 SparkContext，app 中存在多个 job，每触发一次 action 操作就会产生一个 job。这些 job 可以并行或串行执行，每个 job 中有多个 stage，stage 是 shuffle 过程中 DAGSchaduler 通过 RDD 之间的依赖关系划分 job 而来的，每个 stage 里面有多个 task，组成 taskset 有 TaskSchaduler 分发到各个 executor 中执行，executor 的生命周期是和 app 一样的，即使没有 job 运行也是存在的，所以 task 可以快速启动读取内存进行计算

3）hadoop 的 job 只有 map 和 reduce 操作，表达能力比较欠缺而且在 mr 过程中会重复的读写 hdfs，造成大量的 io 操作，多个 job 需要自己管理关系

4）spark 的迭代计算都是在内存中进行的，API 中提供了大量的 RDD 操作如 join，groupby 等，而且通过 DAG 图可以实现良好的容错

220.【简答题】【Spark】Cache 和 pesist 的区别。

cache：缓存数据，默认是缓存在内存中，其本质还是调用 persist

persist:缓存数据，有丰富的数据缓存策略。数据可以保存在内存也可以保存在磁盘中，使用的时候指定对应的缓存级别就可以了。

221.【简答题】【Spark】Spark master 使用 zookeeper 进行 HA 的，有哪些元数据保存在 Zookeeper？

spark 通过这个参数 spark.deploy.zookeeper.dir 指定 master 元数据在 zookeeper 中保存的位置，包括 Worker，Driver 和 Application 以及 Executors。standby 节点要从 zk 中，获得元数据信息，恢复集群运行状态，才能对外继续提供服务，作业提交资源申请等，在恢复前是不能接受请求的。另外，Master 切换需要注意 2 点：

1）在 Master 切换的过程中，所有的已经在运行的程序皆正常运行。因为 Spark Application 在运行前就已经通过 Cluster Manager 获得了计算资源，所以在运行时 Job 本身的调度和处理和 Master 是没有任何关系的。

2） 在 Master 的切换过程中唯一的影响是不能提交新的 Job：一方面不能够提交新的应用程序给集群，因为只有 Active Master 才能接受新的程序的提交请求；另外一方面，已经运行的程序中也不能够因为 Action 操作触发新的 Job 的提交请求。

222.【单选题】【Storm】下列对于 Storm 集群部署描述错误的一项是（ ）

A、Storm 集群依赖于外部的 Zookeeper

B、Storm 在 0.9 版本之前，环境部署必须搭建 ZeroMQ 和 JZMQ 软件

C、supervisor.slot.ports 是用来配置指定多少个 Executor 运行在机器上

D、storm.Zookeeper.servers:是一个为 Storm 集群配置的 Zookeeper 集群的主机列表

答案：C。

解析：supervisor.slot.ports 是用来配置指定多少个 Worker 运行在机器上。

223.【多选题】【Storm】 下列对于 Storm 描述错误的一项是（ ）

A、如果 Nimbus 和 Supervisor 重新启动，正在运行的 Topologies 会停止

B、启动 Storm 进程之前必须启动 Hadoop 与 Zookeeper 进程

C、Storm 是一个快速失败的系统，这意味着这些进程随时都可能因发生错误而停止

D、Nimbus 的作用类似于 Hadoop 中的 JobTracker 的角色

答案：A、B。

解析：如果 Nimbus 和 Supervisor 重新启动，正在运行的 Topologies 会正常运行；启动 Storm 进程之前必须启动 Zookeeper 进程。

224.【单选题】【Storm】 下列关于 storm.yaml 参数解释错误的是（ ）

A、supervisor.slot.ports: 每一台 worker 机器，你用这个配置来指定多少 workers 运行在哪台机器

B、storm.Zookeeper.servers: 这是一个为 Storm 集群配置的 Zookeeper 集群的主机列表

C、javA、library.path: 这是 Storm 使用的本地库（ZeroMQ 和 JZMQ）载入路径

D、storm.local.dir: Nimbus 和 Supervisor 守护程序需要一个本地磁盘目录存储大量状态（像 jars，conf，其它），每台机器都创建这些目录，赋可写权限

答案：D。

解析：storm.local.dir: Nimbus 和 Supervisor 守护程序需要一个本地磁盘目录存储小量状态（像 jars，conf，其它），每台机器都创建这些目录，赋可写权限。

225.【多选题】【Storm】 下列哪些选项是安装 Storm 前所必须安装的？（）

A、Python

B、JDK

C、Zookeeper

D、Shell Script

答案：A、B、C。

226.【判断题】【Flume】Flume 采集数据会丢失吗?

不会

解析：Channel 存储可以存储在 File 中，数据传输自身有事务

227.【单选题】【Storm】 下列哪个选项是 storm.yaml 配置文件中的内容？（ ）

A、storm.zookeeper.servers

B、JAVA_HOME

C、supervisor.slots.ports

D、drpC、servers

答案：B。

228.【单选题】【Storm】 下列对于 Storm 进程启动描述正确的一项是（ ）？

A、启动 Storm 全部进程命令 bin/start-all.sh

B、后台启动 storm nimbus 进程命令 nohup bin/storm nimbus

C、后台启动 storm nimbus 进程命令 bin/storm nimbus &

D、后台启动 storm nimbus 进程命令 nohup bin/storm nimbus &

答案：D。

解析：Storm 中没有自带的 start-all.sh 命令，需要写脚本才能实现这个功能；对于 nimbus 进程后台启动，如果采用 nohup 命令，必须要以 nohup 开头，&结尾的形式。

229.【单选题】【NoSQL】下面选项中哪个选项不是 NoSQL 数据库（ ）

A、Cassandra

B、HBase

C、Hive

D、Redis

答案：C。

解析：Hive 是基于 Hadoop 的一个数据仓库工具,可以将结构化的数据文件映射成一张表,并提供类 sql 语句的查询功能。

230.【单选题】【NoSQL】下列关于 NoSQL 与关系型数据库的比较错误的是（ ）

A、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

B、NoSQL 很难实现横向扩展，纵向扩展的空间也比较有限

C、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

D、RDBMS 都可以很容易实现数据完整性，但 NoSQL 却无法实现

答案：B。

解析：RDBMS 很难实现横向扩展，纵向扩展的空间也比较有限。

231.【编程题】【SQL】每个用户截止到每月为止的最大单月访问次数和累计到该月的总访问次数?

数据如下：数据的字段意义是： 用户、月份、访问量

A,2015-01,5

A,2015-01,15

B,2015-01,5

A,2015-01,8

B,2015-01,25

A,2015-01,5

A,2015-02,4

A,2015-02,6

B,2015-02,10

B,2015-02,5

A,2015-03,16

A,2015-03,22

B,2015-03,23

B,2015-03,10

B,2015-03,11

编程代码：

```
select A、name as aname, A、month as amonth, A、pv as apv,
max(B、pv) as maxpv, sum(B、pv) as sumpv
from
(select A、name, A、month, sum(A、pv) as pv  from exercise01 a  group by A、name, A、month) a
join
(select A、name, A、month, sum(A、pv) as pv  from exercise01 a  group by A、name, A、month) b
on A、name = B、name
where A、month >= B、month
group by A、name, A、month, A、pv;
```

232.【单选题】【MongoDB】下列关于 MongoDB Shell 描述错误的是（ ）

A、查询所有数据库：show dbs

B、查询当前数据库下所有集合：show collections

C、查询第一条数据：dB、<collectionName>.find()

D、删除该集合下所有文档：dB、<collectionName>.remove({})

答案：C。

解析：db.<collectionName>.find()：查询该集合中所有的数据；dB、<collectionName>.findOne()查询该集合中第一条数据。

233.【单选题】【MongoDB】下列关于 MongoDB 常见的封装结构描述错误的是（ ）

A、db.serverStatus()：查看 mongoDB 运行状态信息

B、db.printSlaveReplicationInfo()：查看复制集同步信息

C、db.collection.createIndex()：创建索引

D、db.stats()：查看 mongoDB 运行状态信息

答案：D。

解析：dB、stats()：查看 db 元数据。

234.【简答题】【大数据概念】你对“大数据”一词有什么理解？

大数据是与复杂和大型数据集相关的术语。关系数据库无法处理大数据，这就是为什么使用特殊的工具和方法对大量数据执行操作的原因。大数据使公司能够更好地了解其业务，并帮助他们从定期收集的非结构化和原始数据中获取有意义的信息。大数据还使公司能够根据数据做出更好的业务决策。

235.【简答题】【大数据概念】大数据的五个 V 是什么？

大数据的五个 V 如下：

卷（Volume）-卷表示卷，即以高速率增长的数据量，即以 PB 为单位的数据量

速度（Velocity）－速度是数据增长的速度。社交媒体在增长数据的速度中起着重要作用。

多样性（Variety）－多样性是指不同的数据类型，即各种数据格式，例如文本，音频，视频等。

准确性（Veracity）－准确性是指可用数据的不确定性。由于大量数据带来不完整和不一致，因此会出现准确性。

价值（Value）－价值是指将数据转化为价值。通过将访问的大数据转化为价值，企业可以产生收入。

5 V的大数据

注意： 这是大数据采访中提出的基本且重要的问题之一。如果您看到面试官有兴趣了解更多信息，则可以选择详细解释五个 V。但是，如果有人询问“大数据”一词，甚至可以提及这些名称。

236.【简答题】【大数据概念】告诉我们大数据和 Hadoop 之间的关系。

大数据和 Hadoop 几乎是同义词。随着大数据的兴起，专门用于大数据操作的 Hadoop 框架也开始流行。专业人士可以使用该框架来分析大数据并帮助企业做出决策。

注意： 在大数据采访中通常会问这个问题。 可以进一步去回答这个问题，并试图解释的 Hadoop 的主要组成部分。

237.【简答题】【大数据概念】大数据分析如何有助于增加业务收入？

大数据分析对于企业来说已经变得非常重要。它可以帮助企业与众不同，并增加收入。通过预测分析，大数据分析为企业提供了定制的建议。此外，大数据分析使企业能够根据客户的需求和偏好推出新产品。这些因素使企业获得更多收入，因此公司正在使用大数据分析。通过实施大数据分析，公司的收入可能会大幅增长 5-20％。一些使用大数据分析来增加收入的受欢迎的公司是-沃尔玛，LinkedIn，Facebook，Twitter，美国银行等。

238.【简答题】【大数据认知】您有大数据经验吗？如果是这样，请与我们分享。

由于该问题是主观问题，因此没有具体答案，并且答案取决于您以前的经验。在大数据采访中问这个问题时，采访者想了解您以前的经验，并且还试图评估您是否适合项目要求。

那么，您将如何处理这个问题？如果您以前有经验，请从以前的职务开始，然后慢慢地在对话中添加细节。告诉他们您使项目成功的贡献。这个问题通常 是面试中问到的第二个 或第三个问题。后面的问题基于此问题，因此

请仔细回答。您还应注意不要过度处理以前的工作。保持简单明了。

239.【简答题】【大数据分析】您喜欢好的数据还是好的模型？为什么？

这是一个棘手的问题，但通常在大数据采访中会问到。它要求您在良好的数据或良好的模型之间进行选择。作为候选人，您应该尝试根据自己的经验来回答。许多公司希望遵循严格的数据评估流程，这意味着他们已经选择了数据模型。在这种情况下，拥有良好的数据可能会改变游戏规则。另一种方法是根据良好的数据选择模型。

如前所述，请根据您的经验进行回答。但是，不要说拥有良好的数据和良好的模型很重要，因为在现实生活中很难同时拥有两者。

240.【简答题】【数据采集】如何将非结构化数据转换为结构化数据？

非结构化数据在大数据中非常常见。应将非结构化数据转换为结构化数据，以确保进行正确的数据分析。您可以通过简要区分两者来开始回答问题。完成后，您现在可以讨论将一种形式转换为另一种形式的方法。您也可能会分享实际情况。如果您刚毕业，则可以共享与您的学术项目有关的信息。

通过正确回答此问题，您表示您已了解结构化和非结构化数据的类型，并且具有处理这些数据的实践经验。如果您具体回答该问题，那么您肯定可以破解大数据采访。

241.【简答题】【数据采集】您如何进行数据准备？

数据准备是大数据项目中的关键步骤之一。大数据采访可能涉及基于数据准备的至少一个问题。当面试官问您这个问题时，他想知道您在数据准备过程中采取了哪些步骤或预防措施。

如您所知，需要进行数据准备才能获得必要的数据，然后可以将这些数据进一步用于建模目的。您应该将此信息传达给面试官。您还应该强调将要使用的模型的类型以及选择该特定模型的原因。最后但并非最不重要的一点，您还应该讨论重要的数据准备术语，例如转换变量，离群值，非结构化数据，识别差距等。

242.【简答题】【数仓概念】数据仓库为什么要分层？

将一个复杂的任务分解成多个步骤来完成，每一层只处理单一的步骤，比较简单、并且方便定位问题

规范数据分层，通过的中间层数据，能够减少极大的重复计算，增加一次计算结果的复用性

不论是数据的异常还是数据的敏感性，使真实数据与统计数据解耦开

243.【简答题】【数仓概念】数据仓库的输入数据源和输出系统分别是什么？

输入系统：埋点产生的用户行为数据、JavaEE 后台产生的业务数据。

输出系统：报表系统、用户画像系统、推荐系统

244.【简答题】【数仓概念】ODS 层（原始数据层）的作用？

原始数据层，存放原始数据，直接加载原始日志、数据，数据保持原貌不做处理

245.【简答题】【数仓概念】DWD 层（原始数据层）的作用？

结构和粒度与原始表保持一致，对 ODS 层数据进行清洗（去除空值，脏数据，超过极限范围的数据），也有公司叫 DWI。

246.【简答题】【数仓概念】DWS 层（原始数据层）的作用？

以 DWD 为基础，进行轻度汇总。一般聚集到以用户当日，设备当日，商家当日，商品当日等等的粒度。

247.【简答题】【数仓概念】ADS 层（原始数据层）的作用？

ADS 层，为各种统计报表提供数据，也有公司或书把这层命名为 APP 层、DM 层等。

248.【简答题】【数仓概念】数据集市与数据仓库区别?

数据集市则是一种微型的数据仓库，它通常有更少的数据，更少的主题区域，以及更少的历史数据，因此是部门级的，一般只能为某个局部范围内的管理人员服务。

数据仓库是企业级的，能为整个企业各个部门的运行提供决策支持手段。

249.【简答题】【数仓概念】数仓命名规范？

ODS 层命名为 ods

DWD 层命名为 dwd

DWS 层命名为 dws

ADS 层命名为 ads

临时表数据库命名为 xxx_tmp

备份数据数据库命名为 xxx_bak

250.【简答题】【数仓概念】Tez 引擎优点？

Tez 可以将多个有依赖的作业转换为一个作业，这样只需写一次 HDFS，且中间节点较少，从而大大提升作业的计算性能。

251.【简答题】【数仓概念】什么是实体表和维度表？

实体表，一般是指一个现实存在的业务对象，比如用户，商品，商家，销售员等等

维度表，一般是指对应一些业务状态，编号的解释表。也可以称之为码表，比如地区表，订单状态，支付方式，审批状态，商品分类等等

252.【简答题】【数仓概念】什么是事务型事实表和周期性事实表？

事务型事实表，一般指随着业务发生不断产生的数据。特点是一旦发生不会再变化，一般比如，交易流水，操作日志，出库入库记录等等

周期型事实表，一般指随着业务发生不断产生的数据，与事务型不同的是，数据会随着业务周期性的推进而变化，比如订单，其中订单状态会周期性变化。再比如，请假、贷款申请，随着批复状态在周期性变化

253.【简答题】【数仓概念】什么是同步策略？

数据同步策略的类型包括：全量表、增量表、新增及变化表、拉链表

全量表：存储完整的数据

增量表：存储新增加的数据

新增及变化表：存储新增加的数据和变化的数据

拉链表：对新增及变化表做定期合并

254.【简答题】【数仓概念】什么是实体表同步策略？

实体表：比如用户，商品，商家，销售员等

实体表数据量比较小：通常可以做每日全量，就是每天存一份完整数据。即每日全量

255.【简答题】【数仓概念】什么是维度表同步策略？

维度表：比如订单状态，审批状态，商品分类

维度表数据量比较小：通常可以做每日全量，就是每天存一份完整数据。即每日全量。

说明：

针对可能会有变化的状态数据可以存储每日全量

没变化的客观世界的维度（比如性别，地区，民族，政治成分，鞋子尺码）可以只存一份固定值

256.【简答题】【数仓概念】什么是事务型事实表同步策略？

事务型事实表：比如，交易流水，操作日志，出库入库记录等

因为数据不会变化，而且数据量巨大，所以每天只同步新增数据即可，所以可以做成每日增量表，即每日创建一个分区存储

257.【简答题】【数仓概念】什么是周期型事实表同步策略？

周期型事实表：比如，订单、请假、贷款申请等

这类表从数据量的角度，存每日全量的话，数据量太大，冗余也太大。如果用每日增量的话无法反应数据变化

每日新增及变化量，包括了当日的新增和修改。一般来说这个表，足够计算大部分当日数据的。但是这种依然无法解决能够得到某一个历史时间点（时间切片）的切片数据

所以要用利用每日新增和变化表，制作一张拉链表，以方便的取到某个时间切片的快照数据。所以我们需要得到每日新增及变化量

258.【简答题】【数仓概念】使用范式的根本目的是？

减少数据冗余，尽量让每个数据只出现一次

保证数据一致性

缺点是获取数据时，需要通过 Join 拼接出最后的数据

259.【简答题】【数仓概念】什么是关系建模和维度建模？

关系模型主要应用与 OLTP 系统中，为了保证数据的一致性以及避免冗余，所以大部分业务系统的表都是遵循第三范式的

维度模型主要应用于 OLAP 系统中，因为关系模型虽然冗余少，但是在大规模数据，跨表分析统计查询过程中，会造成多表关联，这会大大降低执行效率

所以把相关各种表整理成两种：事实表和维度表两种。所有维度表围绕着事实表进行解释

260.【简答题】【数仓概念】什么是拉链表，为什么要做拉链表？

拉链表，记录每条信息的生命周期，一旦一条记录的生命周期结束，就重新开始一条新的记录，并把当前日期放入生效开始日期

拉链表适合于：数据会发生变化，但是大部分是不变的。

比如：订单信息从未支付、已支付、未发货、已完成等状态经历了一周，大部分时间是不变化的。如果数据量有一定规模，无法按照每日全量的方式保存。 比如：1 亿用户*365 天，每天一份用户信息。(做每日全量效率低)

261.【简答题】【Sqoop】 Sqoop 导入导出 Null 存储一致性问题？

Hive 中的 Null 在底层是以“\N”来存储，而 MySQL 中的 Null 在底层就是 Null，为了保证数据两端的一致

性。在导出数据时采用--input-null-string 和--input-null-non-string 两个参数。导入数据时采用--null-string 和--null-non-string

262.【简答题】【Sqoop】Sqoop 底层运行的任务是什么？

只有 Map 阶段，没有 Reduce 阶段的任务

263.【简答题】【Sqoop】Hive 中的 Null 在底层是以“\N”来存储，而 MySQL 中的 Null 在底层就是 Null，如何保证数据两端的一致性？

在导出数据时采用--input-null-string 和--input-null-non-string 两个参数。导入数据时采用--null-string 和--null-non-string。

264.【简答题】【推荐算法】您是否会优化算法或代码以使其运行更快？

这个问题的答案应该始终是“是”。现实世界中的性能很重要，它并不取决于您在项目中使用的数据或模型。

面试官也可能想知道您以前是否有代码或算法优化方面的经验。对于初学者而言，这显然取决于他过去从事的项目。经验丰富的候选人也可以相应地分享他们的经验。但是，请诚实对待您的工作，如果您过去没有优化代码，那也很好。只要让面试官知道您的实际经验，您就可以破解大数据面试。

265.【简答题】【大数据概念】解释部署大数据解决方案应遵循的步骤。

以下是部署大数据解决方案的三个步骤 -

（1）资料撷取

部署大数据解决方案的第一步是数据摄取，即从各种来源提取数据。数据源可以是 Salesforce 之类的 CRM，SAP 之类的企业资源计划系统，MySQL 之类的 RDBMS 或任何其他日志文件，文档，社交媒体源等。可以通过批处理作业或实时流来摄取数据。然后将提取的数据存储在 HDFS 中。

部署大数据解决方案的步骤

（2）数据存储

提取数据后，下一步是存储提取的数据。数据可以存储在 HDFS 或 NoSQL 数据库（即 HBase）中。HDFS 存储适用于顺序访问，而 HBase 适合随机读取/写入访问。

（3）数据处理

部署大数据解决方案的最后一步是数据处理。数据通过 Spark，MapReduce，Pig 等处理框架之一进行处理。

266.【简答题】【Flink】Flink 中对窗口的支持包括哪几种?说说他们的使用场景

1)TumblingTimeWindow 假如我们需要统计每一分钟中用户购买的商品的总数，需要将用户的行为事件按每一分钟进行切分，这种切分被成为翻滚时间窗口(TumblingTimeWindow)。翻滚窗口能将数据流切分成不重叠的窗口，每一个事件只能属于一个窗口。

2)SlidingTimeWindow

我们可以每 30 秒计算一次最近一分钟用户购买的商品总数。这种窗口我们称为滑动时间窗口(SlidingTimeWindow)。在滑窗中，一个元素可以对应多个窗口。

3)TumblingCountWindow

当我们想要每 100 个用户购买行为事件统计购买总数，那么每当窗口中填满 100 个元素了，就会对窗口进行计算，这种窗口我们称之为翻滚计数窗口(TumblingCountWindow)，上图所示窗口大小为 3 个。

4)SessionWindow

在这种用户交互事件流中，我们首先想到的是将事件聚合到会话窗口中(一段用户持续活跃的周期)，由非活跃的间隙分隔开。如上图所示，就是需要计算每个用户在活跃期间总共购买的商品数量，如果用户 30 秒没有活动则视为会话断开(假设 rawdatastream 是单个用户的购买行为流)。一般而言，window 是在无限的流上定义了一个有限的元素集合。这个集合可以是基于时间的，元素个数的，时间和个数结合的，会话间隙的，或者是自定义的。Flink 的 DataStreamAPI 提供了简洁的算子来满足常用的窗口操作，同时提供了通用的窗口机制来允许用户自己定义窗口分配逻辑。