**HDFS 中保证 Name Node 高可用性的角色**（Journalnode/ZKFC/Zookeeper）2、**YARN 在集群中的作用正确**：集群资源管理；集群任务调度与管理3、**Yarn服务中不包含角色**：Container4、**外表和托管表正确**：删除外表只会删除 Inceptor 上的元数据不会删除数据文件，删除托管表两者都会被删除5、**分桶表正确**：分桶表通过改变数据的存储分布，对查询起到一定的优化作用6、**inceptor excutor资源配置正确**：Excutor资源配置 fixed 和 ratio 两种模式；Excutor 内核数配置的是每个 excutor所使用的逻辑 core 数量；Excutor内核数和内存配置比例一般为1 core:2G memory7、**场景查询**：'cache' = 'RAM','holodesk.index' = 'Department', 'holodesk.dimension' = 'Sex, Region'8、**Hyperbase全局索引正确：**A核心是倒排表B.全局索引概念是对应 Rowkey 这个“一级”索引 D.全局索引使用 B+树检索数据9、**Hyperbase最小单元**Region10、**Hyperbase正确**：以上都不正确11、**StreamSQL正确**：Stream是数据流；Application是一个或多个streamjob集合12、**交通部门：**选（C）CREATE STREAM traffic\_stream AS SELECT \* FROM original\_stream STREAMWINDOW w1 AS (length '24' hour slide '1' minute);13、**不是Zookeeper功能**：存储大量数据14、**与Zookeeper通信**：Actice ResourceManager15、**flume和sqoop对比错误**：B.flume 主要采集流式数据而 sqoop 主要用来迁移规范化数据 C . flume 和 sqoop 都是分布式处理任务16、**sqoop抽取数据原理错误**：sqoop 抽取数据是个多节点并行抽取的过程，因此 map 的个数设置的越多性能越好17**、sqoop数据转换错误**：--hive-drop-import-delims 用来设置在hdfs生成的文件的存储形式为列存储18、**flume错误**：flume 和 sqoop 功能相似，因此可以相互替代19、**flume不支持的sink**：memory20、**ElasticSearch错误：**ElasticSearch 数据存储在 HDFS 上21、**不属于kafka应用场景：**关系型数据库和大数据平台见的数据迁移22、**TDH认证模式：**A.所有服务使用简单认证模式—所有服务都无需认证即可互相访问B. 所有服务都启用 Kerberos 认证，用户要提供 Kerberos principal和密码(或者keytab)来访问各个服务C. 所有服务都启用 Kerberos 同时 Inceptor 启用LDAP 认证23、**各组件运维页面错误：**通过Resource Manager的8180对YARN上运行的任务进行监控24、**Incepter查看日志：**B查看nceptor server所在节点/var/log/inceptorsql\*/目录下的hive-server2.log 25、**Hadoop组建的场景**全正确26、**不属于管理角色**：Node Manager27、**不属于集群预安装**：配置集群安全模式28、**HDFS正确：**规划 HDFS 集群时，建议 Active NameNode 和 Standby NameNode 分配在不 同机架上29、**有关Yarn描述错误:**NodeManager 负责调度当前节点的所有 ApplicationMaster30、**Spark对比MR，优势不包括**：Spark 可以运行在 YARN 之上而 MapReduce 不能31、**LOAD命令错误：**源数据文件存放于 hdfs 上，通过 load 命令加载数据文件，数据文件将被复制到表目录下32、**tableA10G，tableB100G：**建表时将 tableA 和 tableB 根据 id 字段分相同数量的桶33、**不属于Hyperbase存储模型单位：**region34、**Minor Compact正确：**把多个HFile合成一个35、**stream错误：**定义Derived stream后stream当即根据转换规则进行变形36、**部门AB，设计合理：**每个部门起一个application管理本部门的streamjob37、**Zookeeper服务正确**：它是分布式应用程序协调服务38、**hue实现hdfs：**创建目录、上传文件、直接查看文件、更改权限39、**oozie workflow调度sqoop正确：**B. 确保对应的jdbc 驱动正确上传到hdfs 上C. Sqoop 导入的 hdfs 目录必须前提不存在40、**sqoop参数错误**：--query 是执行 sqoop 操作的必需参数41、**不属于Flume的source类型：**file source42、**ES特性错误：**D以上都错误43、**安装kerberos切换用户：**直接使用kinit用户名称方式进行切换44、Transwarp **Manage错误：**在 Transwarp Manager 上能启动和停止 Transwarp Agent 角色45、**电信设计：**Hyperbase表+全局索引46、安装TDH操作系统：SUSE、Centos、REHL46、**Yarn负责集群资源管理：**Resource Manager47、**MR特点：**自动化并行和分布式计算、出错容忍度高、优先数据本地化计算（选D）48、**Inceptor数据倾斜正确：**C. 导入数据期间格式转换出现错误引起 null 过多，可以通过重新清理数据解决D. 将一起数据倾斜的数据和剩下的数据单独运行，再通过 union 合并的方式解决49、**inceptor日志信息正确：**A. Inceptor server 日志存放于各节点的/var/log/inceptorsql[x]/hive-server.logB. 可以通过 inceptor server 4040 查看 SQL 错误日志C. Excutor 日志存放于 excutor 节点的/var/log/inceptorsql[x]/ spark-excutor.logD. ExcutorGC 日志存放于 excutor 节点的/var/log/inceptorsql[x]/spark-excutor.gc.log

1. **HMaster功能：**A为Region server分配region；D管理用户对table的增删改查操作51、**创建全局索引正确**B.add\_global\_index't1','index\_name’,'COMBINE\_INDEX|INDEXED=f1:q1:9|rowKey:rowKey:

10,UPDATE=true'51、**流处理错误：**D以上都不对52、**Hue对hive正确：**只支持hiveserver2 53、**oozie三个编辑器正确：**C.workflow是最简单的一种工作方式D.coordinator可以包含一到多个workflow54、Guardian功能：用户管理、用户认证、审计、权限管理55、**表存储hyperbase，原表10G，压缩1:3，索引数据量20G，存储空间**：23.33G**HDFS高可靠协调服务的共享存储**JournalNodes**ResourceManager功能错误：**它直接将集群所拥有的资源按需分配给运行在 YARN 上的应用程序56、**词频任务失败原因**：D都有可能57、**maptask数据多合并参数：**A.SET ngmr.partition.automerge = TRUE;B. SET ngmr.partition.mergesize = n;C. SET ngmr.partition.mergesize.mb = m;58、**Hyperbase与Incepter关系：**C.Inceptor可以访问HyperbaseD.两者相辅相成59、**Zookeeper正确**：Zookeeper 通过选举机制确定 leader，有且仅有一个。60、**Hue修改HDFS权限**：A.hdfs相应的权限C.以hdfs用户登陆61、**oozie使用ssh条件**：B.oozie用户可以免密钥登陆C.oozie用户必须要有bash权限62、**sqoop连接关系型数据库命令查看表：**Dsqoop list-tables63、**flume sink设置参数：**org.apache.flume.sink.kafka.kafkaSink64、

**Elasticsearch错误：**主节点（master node）进行集群的管理，只负责集群节点添加和删除65、**不能保证kafka数据可靠性的措施**：kafka无法保证数据可靠性66、**开启LDAP后连接Inceptor命令：**B.beeline -u jdbc:hive2://$ip:10000 -n $username -p $password67、**数据清洗，null换为0：**Orc事务表。

**课程总结：哈希取模在哪些技术中使用过,分别发挥什么作用?**

•MapReduce：Map任务将中间结果写入专用内存缓冲区Buffer，同时进行Partition(先按“key hashcode % reduce task number”对数据进行分区，分区内再按key排序) • Sqoop：从Oracle或DB2导入数据时，利用哈希取模实现数据均匀切片• Inceptor / Hive：利用Select...Distributeby...Sortby(Clusterby)实现数据分桶 • Search / ElasticSearch：将Document分入不同的Shard

**ZooKeeper在哪些技术和产品中使用过?分别起什么作用?**

• HDFS：-NameNode HA:Active NN选举• YARN：-ResourceManger HA: Active RM选举、存储元数据• Kafka：-存储元数据、配置管理、Broker 动态扩展、Broker负载均衡、Controller Leader选举，以及Consumer Group变化时的Rebalance • Hyperbase -HMaster选举、存储元数据入口地址 • SolrCloud

**计算框架与资源管理系统是如何协同工作的?**

• 计算框架 -本质:编程模型 -负责画出分布式作业的执行图纸

• 资源管理框架 -本质:管理和调度系统 -负责按照图纸，将代码转化为基于DAG的任务集合

**温故知新：**1、大数据的基本特征：数据规模巨大、数据类型多样、生成和处理速度极快、价值巨大但密度较低。

3、大数据技术体系大致分为几层?每层包含哪些技术?

数据展现：ECharts、D3、Cboard等。数据分析：Hive, Inceptor, Spark SQL (数据仓库 & SQL引擎)；Impala, ArgoDB, Presto(数据集市)；Spark Mllib, Sophon, Discover (人工智能)；ElasticSerach, Search, Solr (搜索引擎)； Storm, Flink DataStream, Spark Streaming, Slipstream (实时流处理引擎)。通用计算：MapReduce(批处理计算框架)；Spark Core(高性能计算框架)。资源管理：YARN, Mesos(资源管理系统)；DC/OS, Kubernetes, TCOS(容器化集群操作系统)。数据存储与管理： HBase, Hyperbase, Cassandra Redis Mongodb Neo4j, StellarDB(分布式No/New SQL数据库) ；HDFS, Shiva(分布式文件系统）。数据采集：

Sqoop, Transporter (结构化数据 & 数据导入导出)；Flume, Kafka(半结构化 / 非结构化数据 & 日志采集 / 分布式消息队列)。数据源：电子商务、社交网络、智能硬件等。数据分析到数据采集之间用ZooKeeper（分布式协调服务）。

4、Apache Hadoop项目包含哪些子项目?简述一下它们的功能。

HDFS（分布式文件系统）、MapReduce（批处理计算框架）、Spark（高性能计算框架）、YARN（分布式资源管理系统）、Docker（容器引擎）、Kubernetes（容器化集群操作系统）、Hadoop数据仓库、Hive（SQL引擎）、HBase（分布式NoSQL数据库）、ElastisSearch（分布式搜索引擎）

1. Spark包含哪些组件?简述一下它们的功能。Spark Core:基础计算框架(批处理、交互式分析) ；Spark SQL:SQL引擎(海量结构化数据的高性能查询) ；Spark Streaming:实时流处理(微批)；Spark MLlib:机器学习；Spark GraphX:图计算。
2. **HDFS架构中包含哪几种角色?各自承担什么功能?** **Active NameNode(AN):**活动Msater管理节点;管理命名空间;管理元数据;管理Block副本策略;处理客户端读写请求,为DataNode分配任务。**Standby NameNode(SN):**热备Master管理节点;Active NameNode宕机后，快速升级为新的Active;同步元数据，即周期性下载edits编辑日志，生成fsimage镜像检查点文件。**DataNode:**Slave工作节点;存储Block和数据校验和;执行客户端发送的读写操作;通过心跳机制定期向NameNode汇报运行状态和Block列表信息;集群启动时，DataNode向NameNode提供Block列表信息。**Client:**将文件切分为Block;与NameNode交互，获取文件访问计划和相关元数据;与DataNode交互，读取或写入数据;管理HDFS。
3. **为什么HDFS不合适存储大量的小文件?**元数据占用NameNode大量内存空间（每个文件、目录和Block的元数据都要占用150Byte；存储1亿个元素，大约需要20GB内存；如果一个文件为10KB，1亿个文件大小仅有1TB，却要消耗掉20GB内存）磁盘寻道时间超过读取时间。
4. **Block副本的放置策略是什么?如何理解?**副本1:放在Client所在节点 -对于远程Client，系统会随机选择节点；副本2:放在不同的机架节点上；副本3:放在与第二个副本同一机架的不同节点上；副本N:随机选择；节点选择:同等条件下优先选择空闲节点。
5. **HDFS离开安全模式的条件是什么?**Block上报率:DataNode上报的可用Block个数 / NameNode元数据记录的Block个数 • 当Block上报率 >= 阈值时，HDFS才能离开安全模式，默认阈值为0.999
6. **HDFS是如何实现高可用的?**Active NN与Standby NN的主备切换。利用QJM实现元数据高可用。利用ZooKeeper实现Active节点选举。
7. **简述YARN与MapReduce的关系**。 YARN的出现为了处理MapReduce的缺陷（身兼两职:计算框架 + 资源管理系统。它的JobTracker ：既做资源管理，又做任务调度 、任务太重，开销过大 、存在单点故障）yarn是分布式通用资源管理系统，可以让mapreduce只做计算框架一件事，而且可以将JobTracker的资源管理、任务调度功能分离。
8. **为什么要设计ApplicationMaster**这一角色? MapReduce既做了全局的管理，又做了资源的管理；AM做了全局的管理，都交给ResourceManager做撑不住，划分成任务去调度。
9. **Zookeeper在YARN中承担了哪些功能?**  Active节点选举；回复Actice RM的原有状态信息。
10. **在项目实践中，如何部署YARN的ResourceManager、NodeManager和HDFS的NameNode、DataNode**？计算跟着数据走：YARN中的NodeManager（计算节点）和HDFS的DataNode（数据节点）要一一对应的部署。数据的位置暴露给计算框架，可以直接从当地节点拿数据，因为任何一个数据都会送到离他最近的一个节点去。ResourceManager、NodeManager这种管理节点需要独立部署。因为数据节点挂载上去会影响管理节点。

15、**队列在资源调度中起什么作用?**队列的状态，可以使RUNNING或者STOPPED.如果队列是STOPPED状态，那么新应用不会提交到该队列或者子队列。同样，如果root被设置成STOPPED，那么整个集群都不能提交任务了。现有的应用可以等待完成，因此队列可以优雅的退出关闭。

1. **容量调度器与公平调度器的区别是什么?**

容量：提前做预算，在预算指导下分享集群资源空闲资源优先分配给“实际资源/预算资源”比值最低的队列。公平：动态分配资源，无需预先设定资源分配比例。

1. **容量调度器会严格按预设比例分配资源吗?**

不会，保持弹性。弹性分配：空闲资源可以分配给任何队列，当多个队列争用时，会按比例进行平衡。支持动态管理:可以动态调整队列的容量、权限等参数，也可动态增加、暂停队列。

1. **简述公平调度器中队列权重和资源抢占的含义。**

队列权重：当队列中有任务等待，并且集群中有空闲资源时，每个队列可 以根据权重获得不同比例的空闲资源。资源抢占：终止其他队列的任务，使其让出所占资源，然后将资源分配给占用资源量少于最小资源量限制的队列。

1. **简述MR Split与HDFS Block的关系。** 没有关系，Split的划分方式由程序设定，Split与HDFS Block没有严格的对应关系
2. **为什么MapReduce要求输入输出必须是key-value键值对**?从直观的角度讲，其中的key很关键，在MR的Shuffle中是要对key做排序的，以方便进行计算。更深层次，更核心的原因是，MR作为一个通用的计算框架来说，你的数据结构的类型是要通用的，key-value就被选作为通用的数据类型，不管什么方式的数据传输进来都可以被简单的转换为key-value的形式。
3. **简述Shuffle的工作原理。** Map端：-Map任务将中间结果写入专用内存缓冲区Buffer(默认100M)，同时进行Partition和Sort。“keyhashcode%reducetasknumber”对数据进行分区，分区内再按key排序）。--当Buffer的数据量达到阈值(默认80%)时，将数据溢写(Spill)到磁盘的一个临时文件中，文件内数据先分区后排序--Map任务结束前，将多个临时文件合并(Merge)为一个Map输出文件，文件内数据先分区后排序。Reduce端：Reduce任务从多个Map输出文件中主动抓取(Fetch)属于自己的分区数据，先写入Buffer，数据量达到阈值后，溢写到磁盘的一个临时文件中。--数据抓取完成后，将多个临时文件合并为一个Reduce输入文件，文件内数据按key排序
4. **从编程模型的视角，MapReduce有哪些优缺点**?缺点：仅支持Map、Reduce两种语义操作；执行效率低，时间开销大；主要用于大规模离线批处理；不适合迭代计算、交互式计算、实时流处理等场景；优点：不容易出错，比较稳定。

23、 **RDD的“弹性”主要体现在哪里**? 弹性分布式数据集，失效后自动重构RDD分解成partition以后，这些分区在内存中。如果RDD失效的话，可以通过transformation自动进行重构生成（弹性）。这个弹性策略导致Spark的基础很牢固。24、**RDD宽依赖为什么又称为Shuffle依赖**? 宽依赖的依赖关系中子RDD的partition要依赖于所有的父RDD。这就表明所有的父节点的任务必须同时完成之后才能启动子节点的任务，因此形成了一种强依赖关系。这种依赖就是对MapReduce的一种再写，因此必须进行Shuffle。25、**Spark运行模式有几种?Driver的主要功能是什么**?运行模式：抽象、Local、Standalone、YARN（clientcluster）。Driver：一Spark程序有一Driver，一Driver创建一SparkContext，程序的main函数运行在Driver中。负责解析Spark程序、划分Stage、调度任务到Executor上执行。**26、简述Spark的程序执行过程。**生成逻辑计划、生成物理计划、任务调度、任务执行。在Driver中先做逻辑计划，生成RDD之间的关系；然后做物理计划，把RDD分解成partition，并且划分任务，形成有向无环图；之后交付给Scheduler去做调度；最后在Executor中执行任务。26、**DAGScheduler是如何划分Task的**? 根据依赖关系是否为宽依赖，即是否存在Shuffle，将DAG划分为不同的阶段(Stage)；将各阶段中的Task组成的TaskSet提交到TaskScheduler**27、DAGScheduler是如何划分Task的？**根据任务的依赖关系建立DAG。根据依赖关系是否为宽依赖，将DAG划分为不同的阶段Stage。将各阶段中的Task组成的TaskSet提交到TaskScheduler。28、**为什么要对Consumer进行分组?** 为了加快读取速度，多个Consumer可划分为一个组（Consumer Group, CG），并行消费同一个Topic。一个Topic可以被多个CG订阅，CG之间是平等的，即一个消息可同时被多个CG消费。一个CG中可以有多个Consumer，CG中的Consumer之间是竞争关系，即一个消息在一个CG中只能被一个Consumer消费。29、**为什么Kafka分了Topic之后，还要分Partition**? Topic：是Kafka中同一类数据的集合，相当于数据库中的表。Topic是逻辑概念，不必关心数据存于何处。Partition（分区）：分区内消息有序存储。一个Topic可分为多个分区，相当于把一个数据集分成多份，分别存储不同的分区中，Parition是物理概念，每个分区对应一个文件夹，其中存储分区的数据和索引文件。30、**Partition Leader和Follower是如何分工合作的**? 从一个分区的多个副本中选举一个Partition Leader，由Leader负责读写，其他副本作为Follower从Leader同步消息30、**为什么Zookeeper不亲自负责Partition Leader**选举? Kafka Controller Leader负责管理Kafka集群的分区和副本状态，避免分区副本直接在Zookeeper上注册Watcher和竞争创建临时Znode，导致Zookeeper集群负载过重

31、**如何定位Inceptor?它与Hive有什么区别?** • 分布式通用SQL引擎 -支持Slipstream、ArgoDB、Hyperbase和Search -构建星环新一代逻辑数据仓库• 分布式数据仓库系统• 基于Hive和Spark打造  
• 用于离线分析和交互式分析(Holodesk -> ArgoDB)

32、**如何理解Inceptor读时模式**。• 含义:数据写入数据仓库时，不检查数据的规范性，而是在查询时再验证；特点-数据写入速度快，适合处理大规模数据-查询时处理尺度很宽松(弱校验)，尽可能恢复各种错误

33、**分区的目的是什么?分区有几种类型**?**如何将数据导入分区表?**  目的:减少不必要的全表扫描，缩小查询范围，提升查询效率；类型：单值分区:一个分区对应分区键的一个值；-单值静态分区:导入数据时，必须手动指定目标分区；-单值动态分区:导入数据时，系统可以动态判断目标分；范围分区:均需手工指定，不支持将文件直接导入范围分区。 导入： 1.数据预处理要求：文件编码为UTF-8，\n为换行。2.将文件导入表或分区（Load导入）：仅将数据文件移动到表或分区的目录中，不会对数据进行任何处理，如分桶、排序。不支持动态，不建议Load。3.将查询结果导入表或分区（Insert导入）。补充：分区表将数据按分区键的键值存储在表目录的子目录中，目录名为“分区键=键值”。Inceptor只支持TEXT表、ORC表、CSV表和Holodesk表的分区操作。

34、**分桶的目的是什么?如何将数据导入分桶表**? • 含义:按分桶键哈希取模的方式，将表中数据随机、均匀地分发到若干桶文件中• 目的:通过改变数据的存储分布，提升取样、Join等特定任务的执行效率 • 将数据写入分桶表  
-分桶表在创建的时候只定义Schema，且数据写入时系统不会自动分桶，所以需要先人工分桶再写入  
-写入分桶表只能通过Insert，而不能通过Load，因为Load只导入文件，并不分桶  
-如果分桶表创建时定义了排序键，那么数据不仅要分桶，还要排序  
-如果分桶键和排序键不同，且按降序排列，使用Distribute by ... Sort by分桶排序  
-如果分桶键和排序键相同，且按升序排列(默认)，使用Cluster by分桶排序

35、**事件驱动模式与微批模式有什么不同**? 微批(Micro-batch)模式:将Input Stream按时间划分成若干小数据块(Batch)来处理，即在由若干单位时间组成的时间间隔内，将接收的数据放到一个Batch中(Batch的时间长度称为Batch Duration)  
-事件驱动(Event-driven)模式:以单条数据被Input Stream接收为事件，逐条读取并处理

37、**简述一下SteamJob的主要作用**。 对一个或多个Stream进行计算，并将结果写入一张表的任务  
StreamJob是触发StreamSQL执行的Action，一般具有插入结果表语义  
StreamJob主要存储StreamJob Level的配置参数，以及对应的SQL

38、**StreamSQL与普通SQL有什么区别**? DML语句的运行机制不同• 普通SQL:阻塞式运行-提交SQL后，用户需等待SQL执行结束，期间命令被持续阻塞，无法执行其他命令• StreamSQL:背景运行-计算任务持续在后台运行-执行StreamSQL的DML语句会立即返回结果 询结果的输出不同• 普通SQL:查询结果或者显示在Console，或者通过JDBC读取• StreamSQL:用户必须显式地指定查询结果输出到某个地方-后台持续运行的SQL无法直接跟Console交互

39、**Search的数据模型与关系数据库有怎样的对应关系**? index(索引)->table(表)；document(文档)->row(行)；field(字段)->column(列)

40、**Search包含哪几类节点，它们各自负责哪些工作**? 主节点(MasterNode)  
• 负责管理集群内的所有变更，如增删节点、增删索引、分配分片等，不负责文档更新和搜索  
• 每个集群只有一个主节点，默认情况下任何节点都可能被选为主节点  
• 硬件配置:普通服务器(CPU、内存消耗一般) 数据节点(DataNode)  
• 负责存储数据，即文档的增删改查• 分离主节点和数据节点是一个比较好的选择，因为索引和搜索操作会消耗大量资源• 硬件配置:较高配置服务器(主要消耗磁盘和内存) 客户端节点(ClientNode / 路由节点)• 负责路由请求，实现集群访问的负载均衡  
• 集群规模较大时非常有用，协调主节点和数据节点，根据集群状态直接路由请求41、**简述Index、Document、Shard与副本Shard的关系**。• Shard分为主Shard和副本Shard，后者是前者的精确复制，每个Shard可有零个或多个副本• Index的任意一个Document都归属于一个主Shard，主Shard的数量决定了Index的最大数据量• Index建立时就必须明确主Shard数且不能修改，但副本Shard数可以随时修改• 写操作只能被主Shard处理，读操作可同时被主Shard或副本Shard处理 -对于读操作，理论上拥有更多的副本，将拥有更高的吞吐量，但如果只在相同节点数目的集群上增加副本并不能提高性能，因为每个Shard获得的资源会变少，这时需要增加更多的硬件资源来提升吞吐 量

42、**简述Search更新文档的基本流程**。(1)客户端向Node1(路由节点)发送新建、索引或删除文档请求；2)通过文档id确定该文档属于分片0，请求被转发到Node3，因为分片0的主分片在Node3上  
(3)Node3在主分片上执行更新操作43、**为什么可以将Hyperbase表看作是一张四维表**? 因为它首先第一列是RowKey，下是每一行（包含多行），然后第二列是列族，下面包括很多列限定符，再是时间戳，表示存入时间所以是一张四维表。44、**为什么说Hyperbase是一个Key-Value数据库**? 这里面的key并不只是等于RowKey，而是RowKey加列族加列限定符加时间戳共同构成Key，而Value表示这个表里对应的单元格中的值Cell。

45、**简述Table、Region、Store和StoreFile的关系**。一个table由多行组成，而系统将表水平划分(按行)为多个Region，每个Region保存表的一段连续数据，默认每张表开始只有一个Region，随着数据不断写入，Region不断增大，当Region大小超过阀值时，当前Region会分裂成两个子Region，而一个Region由多个Store组成，每个Store存储一个列族，而store由内存中的MemStore和磁盘中的若干StoreFile组成，MemStore是Store的内存缓冲区，数据读写都先访问MemStore，StoreFile是MemStore的磁盘溢写文件，在HDFS中被称为HFile

46、**为什么要进行Region Split和StoreFile Compaction**? StoreFile Compaction  
• 含义:将Store中的全部或部分StoreFile合并为一个StoreFile的过程  
• 目的:减Region Split  
• 含义:根据一定的触发条件和分裂策略，将Region划分为两个子Region的过程  
• 目的:实现数据访问的负载均衡少StoreFile数量，提升数据读取效率

47、**请描述将一个100GB文件以BulkLoad方式写入HyperBase的主要步骤。**

1、抽取:从数据源中抽取数据 -对于MySQL，运行mysqldump命令导出数据。

2、转换:利用MapReduce，将数据转换为HFile文件 -对于TSV或CSV文件，使用HBase ImportTsv工具将其转换成HFile文件 -每个输出文件夹中的每个区域都会创建一个HFile文件 -HDFS中的可用磁盘空间至少为原始输入文件的两倍。例如:对于100GB的mysqldump导出文件，HDFS中至少预留不少于200GB的磁盘空间，可在任务结束后删除原始输入文件。

3、加载:将HFile文件加载到HBase -利用HBase CompleteBulkLoad工具，将HFile文件移动到HBase表的相应目录中，完成加载。

**48.Inceptor\Hyperbase\StreamSQL\Discover\Search的特性和应用场景**

Inceptor：基于Hadoo的数据仓库产品，支持SQL，存储过程，分布式事务；通过表春测试TPC-H/TPC-DS证明产品对性能的极限优化与提升；混合负载管理与SLA管控；场景：统计分析、批处理、交互式统计分析、图计算和图检索。

Hyperbase：完整支持使用SQL进行高并发业务；支持建立全局/二级索引；半/非结构化数据处理平台，支持对象存储；场景：海量数据存储、高并发操作、数据随机读写操作、数据强一致性。

Search：大规模数据全文检索引擎；支持使用标准SQL扩展支持全文检索；支持混合存储模型，可以利用SSD存储加速；场景：

Discover：机器学习与数据挖掘平台；支持SQL/R/Python等开发接口；提供分布式机器学习算法；提供丰富的行业模版。场景：分布式统计学习恶化机器学习算法库。

StreamSQL：StreamSQL的计算运行于流计算引擎Transwarp Slipstream之上，该引擎混合了事件驱动和微批处理，因此既可以支持有低延迟需求的任务也可以处理高吞吐任务，能够应对不同类型业务。

**49.Yarn的调度策略有哪几种，特点是什么？**

FIFO Scheduler（先进先出调度器）：（策略）将所有任务放入一个队列，先进队列的先获得资源，排在后面的任务只有等待。(缺点)－资源利用率低，无法交叉运行任务。－灵活性差。

Capacity Scheduler（容量调度器）：(核心思想)：提前做预算，在预算指导下分享集群资源。(调度策略)：－集群资源由多个队列分享。－每个队列都要预设资源分配的比例（提前做预算）。－空闲资源优先分配给“实际资源/预算资源”比值最低的队列－队列内部采用FIFO调度策略。(特点):－层次化的队列设计：子队列可使用父队列资源。－容量保证：每个队列都要预设资源占比，防止资源独占。－弹性分配：空闲资源可以分配给任何队列，当多个队列争用时，会按比例进行平衡。－支持动态管理：可以动态调整队列的容量、权限等参数，也可动态增加、暂停队列。－访问控制：用户只能向自己的队列中提交任务，不能访问其他队列。－多租户：多用户共享集群资源。

Fair Scheduler（公平调度器）：（调度策略)：－多队列公平共享集群资源。－通过平分的方式，动态分配资源，无需预先设定资源分配比例。－队列内部可配置调度策略：FIFO、Fair（默认）。

资源抢占+队列权重。