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

50、**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'tl','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）

5、**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。

6、**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。

7、**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，不符合 HDFS 的设计

8、**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。

9、**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

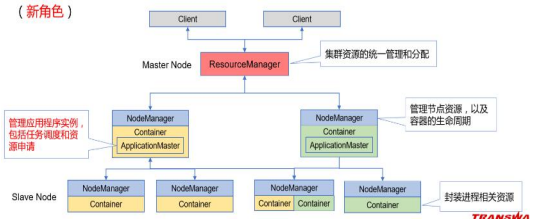
10、**HDFS 是如何实现高可用的?**Active NN 与 Standby NN 的主备切换。利用 QJM 实现元数据高可用。利用 ZooKeeper 实现 Active 节点选举。

11、**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。

12、**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。对于一个集群来说作业是成千上万的，都由 ResourceManager 来管理全部的资源任务太困难，所以创建了 ApplicationMaster 来管理并执行作业。这个角色的设计充分体现了资源管理的分治的思法。管理应用程序实例；向 ResourceManager 申请任务执行所需的资源；任务调度和监管。

13、**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。

14、**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。

16、**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。

17、**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。

18、**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。

19、**简述 MR Split 与 HDFS Block 的关系。** 没有关系，Split 与 HDFS Block 没有严格的对应关系。Split（切片）是逻辑概念，划分方式主要由程序设定，和 Block 没有关系，但是为了便于从 HDFS 中取数据，所以默认一个 Block 大小等于一个 Split 大小。Block 是物理切块。

20、**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。

21、**★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

22、**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?定位：**用于数据仓库和交互式分析的大数据平台软件；• 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)

区别：与 Apache Hive 相比，数据分析处理速度有显著提升。 补充**特点**：Hadoop 领域对 SQL 支持最完善；支持完整分布式事务处理 MVCC；优异的大数据处理和分析性能；提供便捷的 SQL、PL/SQL 开发调试辅助工具 Waterdrop。

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

补充：• 与分区键不同，分桶键必须是表结构中的列• 分桶键和分桶数在建表时确定，不允许更改• ORC 事务表必须分桶• 每个桶的文件大小应在 100~200MB 之间（ORC 表压缩后的数据）• 先分区后分桶。

35、**事件驱动模式与微批模式有什么不同**? 1.相比微批模式，事件驱动模式的延迟更低，在延迟敏感的场景中表现更佳。2.微批秒级，事件驱动毫秒级 3.**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

36、**两种处理模式下的窗口变形有什么不同？微**：对一个时间窗口内的多个 Batch 进行计算得到新 Batch 的过程。•Window Stream：通过窗口变形得到的 Derived Stream。•两个重要参数：Length 和 Slide，Length 窗口持续时间，Slide 两相邻窗口间隔时间。Length 和 Slide 必须是 Batch Duration 的倍数。**事件**：对一个时间窗口内的多条数据进行计算得到新数据的过程。

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•要让 StreamSQL 执行计划，需要有相应的 Action 操作来触发 StreamJob。一个 StreamJob 启动时，StreamSQL 会为每一个 InputStream 启动一组称为 Receiver 的任务来接收数据，接收来的数据经过一系列 Derived Stream 的变形最终被插入一张表，供用户查询。

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**? • **索引上**：与关系数据库的索引不同，这里是指 Search 的数据对象。•**数据对象**：Search: index 索引/ document 文档/ field 字段。关系：Table 表/row 行/column 列 •**map 映射上**：相当于关系数据库中的表结构定义（Schema）。

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，如果成功了，Node3 将请求并行转发到 Node1 和 Node2 的副本分片上，一旦所有副本分片都报告同步成功，Node3 将向 Node1 报告更新成功，最后 Node1 向客户端报告成功。

43、**为什么可以将 Hyperbase 表看作是一张四维表**? **四维表**：RowKey | 列族 | 列限定符 | 时间戳；二维表：RowKey | 列 这四维有：RowKey，列族，列限定符，时间戳。这四维被拍扁的条件是，列和列限定符合二为一变成列，同时时间戳被忽略掉，只更新最新数据。就变成了二维表：RowKey 和列

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。1.按 Key 的字典序顺序存储。2.主要通过 Key 实现数据的增删改查，以及扫库操作。

45、**简述 Table、Region、Store 和 StoreFile 的关系**。**Table**：Hyperbase 以“表”为单位组织数据。表由多行组成。**Region:**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 大小超过阈值时，当前 Region 会分裂成两个子 Region。**Store:**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，。**StoreFile:**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。当 Store 中的 StoreFile 数量超过阈值时，HRegionServer 会将若干小 StoreFile 合并为一个大 StoreFile。当 Region 中最大 Store 的大小超过阈值时，HRegionServer 会将其等分为两个子 Region。Client 读取数据时，先找 MemStore，再找 StoreFile。

46、**为什么要进行 Region Split 和 StoreFile Compaction**? **StoreFile Compaction**

Region Split：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。**方法**：利用 Middle Key 将当前 Region 划分为两个等分的子 Region。**条件**：当 Region 中最大 Store 的大小超过阈值时，触发 Region Split。

StoreFile Compaction：将 Store 中的全部或部分 StoreFile 合并为一个 StoreFile 的过程。**目的**：减少 StoreFile 数量，提升数据读取效率。**条件**：当 Store 中的 StoreFile 数量超过阈值，触发 StoreFile Compaction。

47、**请描述将一个 100GB 文件以 BulkLoad 方式写入 HyperBase 的主要步骤。（Hbase BulkLoad 的基本过程）**

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。

50. SQL BulkLoad 的操作步骤。

第 1 步：将数据集上传至 HDFS• 第 2 步：为 HDFS 中的数据集创建 Inceptor 外表。• 第 3 步：对外表（第 2 步创建）预分 Region，获取 Split Key。第 4 步：在 Inceptor 中创建 Hyperdrive 表（HBase 表的二维映射表），利用 Split Key（第 3 步获取）对 HBase 表预分 Region。• 第 5 步：在 Inceptor 中使用 SQL BulkLoad 语句，将外表中的数据导入 Hyperdrive 表。

其他重点：

Sqoop 是一个主要在 Hadoop 和关系数据库之间进行批量数据迁移的工具。

Flume：Flume 是一个分布式海量数据采集、聚合和传输系统。

Event：事件，最小数据传输单元，由 Header 和 Body 组成。Agent：代理，JVM 进程，最小运行单元，由 Source、Channel、Sink 三个基本组件构成，负责将外部数据源产生的数据以 Event 的形式传输到目的地。映射关系：1 个 Source 多个 Channel，1 个 Channel 多个 Sink，1 个 Sink1 个 Channel。

分布式消息队列 Kafka

一个 Topic 可分为多个 Partition，仅保证同一分区内消息有序存储，不保证 Topic 整体有序。
Kafka 索引：偏移量和时间戳。