ch1.大数据技术综述

1.大数据的基本特征是什么?

数据规模大,数据类型多样,生成和处理速度极快,价值巨大但密度较低。

2.Hadoop 经历了几个发展阶段，各有什么特点?

前 Hadoop 时代：Hadoop 时代：后 Hadoop 时代。

3.大数据技术体系大致分为几层？每层包含哪些技术?

大致分为 7 层,数据展现(ECharts,D3,Cboard);数据分析(数据仓库,数据集市,搜索引擎,SQL 引擎,实时流处理引擎,人工智能);通用计算(批处理计算框架,高性能计算框架);资源管理(资源管理系统,容器化集群操作系统);数据存储与管理(分布式文件系统,分布式 No/New SQL 数据库);数据采集(结构化数据&数据导入导出,非结构化/半结构化数据&日记采集/分布式消息队列);数据源(电子商务、社交网络、智能硬件)

4.Apache Hadoop 项目包含哪些子项目？简述一下它们的功能。

【分布式文件系统 HDFS】数据存储和管理。高容错;高可用;高扩展;简单一致性模型;流式数据访问;大规模数据集;构建成本低且安全可靠**【批处理计算框架 MapReduce】**面向批处理的分布式计算框架。分而治之,分布式计算。移动计算,而不是移动数据。高容错;高扩展;适用于海量数据的离线批处理;降低了分布式编程的门槛**【高性能计算框架 Spark】**计算高效;通用易用;运行模式多样**【分布式资源管理系统 YARN】**资源管理和作业调度。通用;高可用;高扩展**【容器引擎 docker】**打包应用及依赖包到一个可移植的容器中,然后发布到任意一台 Linux 上**【容器化集群操作系统 Kubernetes】**容器化集群管理引擎、生产级容器编排工具**【Hadoop 数据仓库】**企业决策支持**【SQL 引擎 Hive】**对海量结构化数据进行高性能 SQL 查询。提供类 SQL 查询语言;支持命令行或 JDBC/ODBC;提供灵活的扩展性;提供复杂数据模型、扩展函数、脚本等**【分布式 NoSQL 数据库 HBase】**列式存储:用于半结构化、非结构化数据。高并发;高可用;高扩展;海量存储**【分布式搜索引擎 ElastisSearch】**基于 lucene 实现全文数据的快速存储、搜索和分析，处理 PB 级以上数据，强扩展性。

5.spark 包含哪些组件？简述一下它们的功能。

core(基础计算框架-批处理,交互式分析) SQL(SQL 引擎-海量结构化数据的高性能查询) streaming(实时流处理-微批) MLlib(机器学习) GraphX(图计算)

ch2.分布式文件系统 HDFS

1.HDFS 架构中包含哪几种角色？各自承担什么功能?

【Active NameNode】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;集群中唯一【Standby NameNode】AN 宕机后快速升级为 active,同步元数据,即周期性下载 edits,生成 fsimage【NameNode 元数据文件】edits 编辑日志文件,fsimage 元数据检查点镜像文件【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,通过心跳机制周期向 NameNode 汇报运行状态和 block 列表信息,集群启动时向 NameNode 提供 block 列表信息【block 数据块】HDFS 最小存储单元,若一个 block 大小小于设定值不会占用整个块空间,默认 3 个副本【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 【系统架构：Mater/Slave】

2.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，磁盘寻道时间超过读取时间。

3.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,副本 2 放在不同的机架节点,副本 3 放在与副本 2 在同一机架的不同节点,副本 N 随机选择,在同等条件下优先选择空闲节点。

4.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，不建议手动强制退出。
【安全模式是 HDFS 确保 block 数据安全的一种保护机制。HDFS 只接收读数据请求，而不接收写入、删除、修改等变更操作。AN 启动时 HDFS 进入安全模式。触发原因：namenode 重启或磁盘空间不足、block 上报率低于阈值、datanode 无法正常启动、日志中出现严重异常、用户操作不当。故障排查：找到 datanode 不能正常启动的原因，重启 datanode；清理 namenode 磁盘。】

5.HDFS 是如何实现高可用的?

AN 与 SN 的主备切换、利用 QJM 实现元数据高可用(QJM 机制、QJM 共享存储系统)、利用 ZooKeeper 实现 active 节点选举。[JouralNode(共享存储) ZKFC Zookeeper]

6.HDFS 的缺点?

不适合低延迟数据访问；不适合大量小文件存储；不支持并发写入；不支持文件随机修改。

7.元数据存储。

内存元数据(NameNode),文件元数据(edits 编辑日志文件+fsiamge 元数据镜像检查点文件)。

ch3.分布式资源管理系统 YARN

1.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。YARN 提供 MapReduce 的 ApplicationMaster 实现。

2.为什么要设计 ApplicationMaster 这一角色。

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。职责:向调度器索要适当的资源容器,运行任务,跟踪应用程序的状态和监控它们的进程,处理任务的失败原因。

3.ZooKeeper 在 YARN 中承担了哪些功能?

active 节点选举、恢复 Active RM 的原有状态信息。

4.在实践中如何部署 YARN 的 ResourceManager、NodeManager 和 HDFS 的 NameNode、DataNode?

1)HDFS 集群:负责海量数据的存储,集群中的角色主要有 NameNode /DataNode /SecondaryNameNode。2)YARN 集群,负责海量数据运算时的资源调度，集群中的角色主要有 ResourceManager /NodeManager

5.队列在资源调度中起什么作用?

将需要调度的资源放在队列中,进行不同资源调度策略时对不同队列中的资源可以进行不同的处理。

6.容量调度器与公平调度器的区别是什么?

容器调度器的每个队列都要预设资源分配的比例（提前做预算），而公平调度器通过平分的方式，动态分配资源，无需预先设定资源分配比例。

7.容量调度器会严格按预设比例分配资源吗?

容量调度器是弹性分配，空闲资源可以分配给任何队列，多个队列争用时会按比例进行平衡；支持动态管理，可以动态调整队列容量权限等参数，也可动态增加或暂停队列。

8.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。

9.YARN 里的角色。

【ResourceManager】master 集群资源的统一管理和分配【NodeManager】slave 管理节点资源,以及容器的生命周期【ApplicationMaster】管理应用程序实例,包括任务调度和资源申请【container】封装进程相关资源,是 YARN 中资源的抽象。 【系统架构：Mater/Slave】

ch4.分布式计算框架 MapReduce & Spark

1.简述 MR Split 与 HDFS block 的关系。

split 的大小默认等于 block 大小。split 划分方式由程序设定,split 与 HDFS block 没有严格对应关系。

2.为什么 MapReduce 要求输入输出必须是 key-value 键值对?

MapReduce 框架只操作键值对<key, value>，因此这个框架中任务的输入和输出都是键值对形式。

3.简述 shuffle 的工作原理。 [避免和减少 shuffle 是 MapReduce 程序调优的重点 哈希取模]

【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（先按“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?

优点：1)高容错:任务失败，自动调度到其他节点重新执行 2)高扩展:计算能力随着节点数增加，近似线性递增 3)适用于海量数据的离线批处理 4)降低了分布式编程的门槛

缺点：1) OLAP:要求毫秒或秒级返回结果 2)流计算输入数据集是动态的，而 MapReduce 是静态的 3) DAG 计算:多个任务之间存在依赖关系，后一个的输入是前一个的输出，构成有向无环图 DAG。每个 MapReduce 作业的输出结果都会落盘，造成大量磁盘 IO，导致性能非常低下 4)仅支持 map 和 reduce 两种操作。不适合迭代计算、交互式计算、实时流处理等场景。执行效率低，时间开销大。

5.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。

6.RDD 宽依赖为什么又称为 shuffle 依赖?

子 RDD 的部分或全部分区数据丢失或损坏,从所有父 RDD 分区重新计算,必须进行 shuffle。
【窄依赖-父 RDD 中的分区最多只能被一个子 RDD 的一个分区使用，例 map,filter,union。窄依赖-子 RDD 分区依赖父 RDD 的所有分区，例 groupByKey,reduceByKey,sortByKey。】

7.spark 运行模式有几种？driver 的主要功能是什么?

local 模式、standalone 模式、YARN/Mesos 模式。driver 负责解析 spark 程序、划分 stage、调度任务到 executor 上执行。一个 spark 程序有一个 driver，一个 driver 创建一个 SparkContext，程序的 main 函数运行在 driver 中。【SparkContext 负责加载配置信息，初始化运行环境。Executor 负责执行 driver 分发的任务。Task 是 spark 运行的基本单位。】【local 单机运行，测试。standalone 是 spark 集群独立运行，不依赖第三方资源管理系统，Master/Slave，driver 在 worker 中运行，master 只负责集群管理，zookeeper 负责 masterHA 避免单点故障。yarn-client 用于交互和测试，yarn-cluster 用于生产环境】

8.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。

9.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，即是否存在 shuffle，将 DAG 划分为不同的阶段，将各阶段中的 Task 组成的 TaskSet 提交到 TaskScheduler。
TaskScheduler:负责 application 的任务调度,重新提交失败的 task,为执行速度慢的 task 启动备用 task。

ch5.分布式 ETL 工具 Sqoop

定义：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。面向大数据集的批量导入导出(将输入数据集分为 N 个切片，然后启动 N 个 map 任务并行传输)，支持全量、增量两种传输方式。提供多种 Sqoop 连接器：内置(RDBMS,JDBC)、第三方(数据仓库,NoSQL 数据库)。

并发度控制：数据导入的性能瓶颈;数据导入的性能可按每个 map 任务的处理速度 5~10MB/s 做估算;map 个数并非越多越好，过多的 map 会导致 RDBMS 发生 IO 抢占，反而降低整体性能;RDBMS 的导出速度控制在 60~80MB/s;通过 query 人工均匀切分数据。

ch6.分布式数据采集工具 Flume

定义：是一个分布式海量数据采集、聚合和传输系统。特点：基于事件的海量数据采集;数据流模型是 source→channel→sink;事务机制是支持重读重写，保证消息传递的可靠性;内置丰富插件是轻松与各种外部系统集成;高可用是 agent 主备切换;Java 实现是开源，优秀的系统设计。

基本概念：[**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。source 负责对接各种外部数据源，将采集到的数据封装成 event，然后写入 channel。channel 是 event 暂存容器，负责保存 source 发送的 event，直至被 sink 成功读取。sink 是负责从 channel 读取 event，然后将其写入外部存储，或传输给下一阶段的 agent。映射关系：一个 source→多个 channel，一个 channel→多个 sink，一个 sink→1 个 channel。

架构：[**单层架构**]优点：架构简单，使用方便，占用资源较少。缺点：如果采集的数据源或 agent 较多，将 event 写入到 HDFS 会产生很多小文件。系统安全性较差。数据源管理较混乱。外部存储升级维护或发生故障，需对采集层的所有 agent 做处理，人力成本较高，系统稳定性较差。
[**多层架构**]优点：各类日志数据分层处理，架构清晰，运维高效，降低人工误操作风险。避免产生过多小文件，提高系统稳定性和处理能力。对外不会暴露系统关键信息，降低攻击风险，显著提升安全性。各关联系统易于升级维护。缺点：部署相对复杂，占用资源较多。

ch7.分布式消息队列 Kafka

1.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，并行消费同一个 topic。

2.为什么 Kafka 分了 topic 之后，还要分 partition?

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,然后分区可以设置多个副本,副本存储在不同的 Broker 中,这样就可以避免 Kafka 早期版本没有 Replication 概念，一旦某个 Brocker 宕机，其上的分区数据就可能丢失，这样的错误。

3.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。partition follower 定期从 leader 同步数据，partition leader 挂掉后，Kafka controller leader 从 ISR 中选择一个 follower 作为新的 leader。

4.为什么 ZooKeeper 不亲自负责 partition leader 选举?

通过 Zookeeper，从 Kafka 集群中选举出一个 Broker 作为 Kafka Controller Leader。Kafka Controller Leader 负责管理 Kafka 集群的分区和副本状态，负责 Partition Leader 的选举。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。

5.Kafka 特性。Kafka 应用场景。Kafka 索引。

消息持久化;高吞吐;高容错;易扩展;同时支持离线、实时数据处理。异步通信;应用解耦;峰值处理。为提高消息写入和查询速度，为每个 partition 创建索引。偏移量索引;时间戳索引。稀疏存储。

6.Kafka 高可用。

多分区多副本;双层选举(Kafka Controller Leader 选举，Kafka Partition Leader 选举)

ch11.分布式 SQL 引擎 inceptor 【executor 是资源调度的基本单元】

1.如何定位 inceptor？它与 Hive 有什么区别?

定位:用于离线分析和交互式分析;分布式数据仓库系统;分布式通用 SQL 引擎;基于 Hive 和 Spark 打造。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。【场景:统计分析;批处理;交互式统计分析;图计算和图检索】

2.如何理解 inceptor 读时模式。

含义：数据写入数据库时，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，尽可能恢复各种错误。

3.分区的目的是什么？分区有几种类型呢？如何将数据导入分区表?

目的：减少不必要的全表扫描，提升查询效率。
含义：将表按照某个或某几个字段(分区键)划分为更小的数据集。
类型：单值分区：静态、动态分区；范围分区。单值静态：必须手动指定目标分区；单值动态：系统动态判断目标分区，动态分区在静态分区建完之后；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 支持 TEXT 表、ORC 表、CSV 表和 Holodesk 表的分区操作。

4.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。
作用：数据划分-随机均匀；数据集合-key 相同的数据在同一个桶中。提高 join 查询和取样效率。
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。（先分区再分桶）
导入：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。数据写入:分桶表在创建的时候只定义 Schema，数据写入时系统不自动分桶，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：•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，否则必须重新分桶•Inceptor 支持 TEXT 表、ORC 表、ORC 事务表(必须分桶)、CSV 表和 Holodesk 表的分桶操作。
存储方式:在表或分区目录下,每个桶存储为一个文件;桶文件的大小应控制在 100~200MB 之间(ORC 表压缩后);如果桶文件小于 HDFSblock,那么一个桶对应一个 block,否则会存储在多个 block 中。

5.table 分类。

表的元数据存储在 Metastore 中,表的实际数据存储在 HDFS,Hyperbase 和 Search 中。
按所有权分类：托管表(内表):删除内表时,会同时删除表数据,以及 Metastore 中的元数据。有控制权。

外表:删除外表时,不会删除表数据,但会删除 Metastore 中的元数据。系统不具有完全控制权。
按存储格式分类:Text 表、ORC 事务表、Holodesk 表、Hyperbase 表、ES 表

ch12.实时流处理引擎 Slipstream

1.事件驱动模式与微批模式有什么不同?

input stream 输入流
事件驱动模式：以单条数据被 input stream 接收为事件，逐条读取并处理。
微批模式：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。
derived stream 衍生流
微批模式：a)含义：stream 变形是从已有 batch 计算得到新 batch 的过程。b)单 batch 变形：对 stream 中单个 batch 进行计算得到新 batch 的过程。c)窗口变形(多 batch 变形)：对一个时间窗口内多个 batch 进行计算得到新 batch 的过程。
事件驱动模式：a)含义：每得到一条数据就对其进行变形，得到 derived stream。b)但数据变形：对 stream 中单条数据进行计算得到新数据的过程。c)窗口变形(多数据变形)：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。
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。可从容对不同延时级别的业务场景，微批→秒级，事件驱动→毫秒级。通过参数配置和 SQL 改写切换两种流处理模式。

2.两种处理模式下的窗口变形有什么不同?

事件驱动模式是多数据变形，微批模式是多 batch 变形。

3.简述一下 StramJob 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。一个 streamJob 启动时，StreamSQL 会为每一个 input stream 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。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。合理使用 application(一组业务逻辑相关的 StreamJob 集合)划分 StreamJob 开源实现资源的共享和隔离。

4.StreamSQL 与普通 SQL 有什么区别?

1)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。
2)查询结果的输出不同。【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。查询结果通常会插入到表中。

5.背景。

批处理：批量处理;调度延时;处理延时。流处理：流式处理;低延时(开发难;调试难;运维难)。
批处理计算框架：MapReduce,SparkCore,Inceptor,FlinkDataset
流式计算:事件 StormCore,Slipstream,FlinkDataStream 微批 SparkStreaming,Slipstream,StormTrident

6.Slipstream 基础。

定位：融合事件驱动与微批处理的实时流计算引擎。分布式流式 SQL 引擎。
特点：微批模式和事件驱动模式的一体化。支持分布式流式 SQL。强大的优化器提升性能。极高的易用性。产品化程度高。迁移成本低。

7.StreamSQL 窗口。

流应用通常会对一个窗口(时间间隔)内的数据做多表关联、聚合或统计。非窗口计算&窗口计算。
窗口类型：滑动窗口(前后窗口之间有重叠)、跳动窗口(前后窗口之间无重叠)。
窗口切分方式：系统时间切分(默认)、事件时间切分(优先级高,灵活性高,多格式支持)。

ch13.分布式搜索引擎 search

1.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)
type 是 index 的逻辑分类，不映射为关系数据库中的数据对象。

2.search 包含哪几类节点，它们各自负责哪些工作?

节点:一个运行中的 ElasticSearch 实例
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。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。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。

3.简述 index、document、shared 与副本 shard 的关系。

document 保存在 shard 中。index 是逻辑概念，shard 是物理概念，创建 index 时会指定划分为一个或多个 shard，然后分布到集群的各节点中，document 通过哈希取模的方式分配到不同的 shard 中。副本 shard 是主 shard 的精确复制，每个 shard 可有零个多个副本。index 是逻辑概念，shard 是物理概念。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 数可由随时修改。
shard(分片)是 search 的数据存储单元，是数据的容器，不可分割但可收缩。当集群规模扩大或缩小时，系统会自动迁移分片，使数据均匀分布。

4.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。
读取文档：1)客户端向 node1(路由节点)发送读取请求 2)node1 通过文档 id 确定文档属于分片 0，分片 0 的副本存储于所有的三个节点上，这种情况下它将请求转发到 node2。3)node2 将目标温度返回给 node1，然后将文档返回给客户端。

5.search 简介。

基于 ElasticSearch 的大规模分布式搜索引擎。特点：分布式实时存储;分布式实时搜索;SQL 引擎与搜索引擎相融合;高扩展。场景：文档数据库;日志分析与监控;舆情分析;搜素引擎。

ch14.分布式 NewSQL 数据库 Hyperbase

1.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

2.为什么说 Hyperbase 是一个 key-value 数据库?

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

3.简述 table、region、store 和 StoreFile 的关系。

系统将 table 水平划分(按行)为多个 region，每个 region 保存表的一段连续数据。一个 region 由多个 store 组成，每个 store 存储一个列族。store 有内存中的 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。client 读取数据时，先找 MemStore 再找 StoreFile。

4.为什么要进行 Region Split 和 StoreFile Compaction?

StoreFile Compaction：将 store 中的全部或部分 StoreFile 合并为一个 StoreFile 的过程，目的是减少 StoreFile 数量，提升数据读取效率。当 store 中的 StoreFile 数量超过阈值时触发 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，目的是实现数据访问的负载均衡，方法是利用 MiddleKey 将当前 region 划分为两个等分的子 region，当 region 中最大 store 的大小超过阈值时触发 Region Split。
HLog:以 WAL 方式写数据时产生的日志文件。目的是 HRegionServer 意外宕机时的数据恢复。每个 server 维护一个 HLog。先写 HLog，再写 MemStore，最后写 StoreFile。定期删除 HLog 过期数据。

5.简述 HBase BulkLoad 的基本过程。

含义：由于 HBase 中数据以 HFile 文件的形式存储与 HDFS，所以绕过 HBaseAPI，先预分 region，再将数据加工成 HFile 文件，并加载到 HBase 中，从而完成大规模数据的快速入库。
优点：极大提高入库效率；不占用 HRegionServer 资源，显著减轻 Hyperbase 集群的写入压力；利用 MapReduce 加工 HFile 文件，非常高效；提高 job 运行速度，降低 job 执行时间。
【抽取】从数据源中抽取数据(对于 MySQL，运行 mysqldump 命令导出数据)。【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)。【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)
基本流程：1-3 步：对数据集预分 region，获取 split key。4.根据 split key 创建 Hyperdrive 表，预分 region。5.将数据集转换为 HFile 文件，加载到 HBase。为 index 数据划分 region & rebuild index。

6.Hyperbase 简介。

高可靠;高性能;高并发;可伸缩;实时读写;面向列的分布式 NewSQL 数据库。基于 HBase 实现。列式存储。key-value 数据库。采用 HDFS 为文件存储系统。
特点:线性扩展;高可用;[高并发操作;实时随机读写;数据强一致性;海量数据存储]**场景**。
表的特点:数据规模大;无模式;面向列族;稀疏;数据多版本;数据无类型。
系统架构：HMaster 管理表的创建删除修改,系统在运行过程中动态添加删除 HRegionServer;为 server 分配 region;管理元数据;不处理 client 数据读写请求。HRegionServer 处理 client 数据读写请求;管理 region split;管理 storeFile Compaction。【Master/Slave】 zookeeper 实现 HMaster 高可用;监控 server 的上下线信息并通知 HMaster;存储元数据的寻址入口。client 通过接口访问 Hyperbase;为了加快数据访问速度，将元数据缓存在 client cache 中。
•Region 是 HBase 分布式存储的最小单元。

Q1 哈希取模在那些技术中使用过，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。
Sqoop – 从 Oracle 或 DB2 导入数据时，利用哈希取模实现数据均匀分片。
Inceptor / Hive – 利用 select…Distributeby…Sortby(Clusterby)实现数据分桶
Search / ElasticSearch – 将 document 分入不同的 shard

Q2 ZooKeeper 在哪些技术和产品中使用过? 分别起什么作用? 【配置管理、集群管理、分布式锁】
HDFS – NameNode HA : Active NN 选举。YARN – ResourceManager 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; consumer group 变化时的 rebalance。Hyperbase – Hmaster 选举、存储元数据入口地址。SolrCloud

Q3 计算框架与资源管理系统是如何协同工作的?

计算框架 – 本质是编程模型。负责画出分布式作业的执行图纸。
资源管理框架 – 本质是管理和调度系统。负责按照图纸，将代码转化为基于 DAG 的任务集合。

Q4 分治思想主要体现在哪些方面? 分别有哪些具体应用? 架构层面、计算层面、数据层面

Q5 描述一个 100GB 文件写入 HDFS 的整个过程。

1)客户端发送创建文件指令给分布式文件系统 2)文件系统告知 namenode (检查权限，查看文件是否存在。EditLog 增加记录。返回输出流对象) 3)客户端往输出流中写入数据,分成一个个数据包 4)根据 namenode 分配,输出流往 datanode 写数据(多个 datanode 构成一个管道 pipeline,输出流写第一个,后面的转发) 4)每个 datanode 写完一个块后，返回确认信息 5)写完数据，关闭输出流 6)发送完成信号给 namenode

Q6 假设集群的每个节点初始有 6 块硬盘，运行一段时间后，每个节点又加了 4 块新硬盘，为了使数据在所有硬盘上分布均匀，能否通过 hdfs balancer 达到效果，为什么? 并列出能达到效果的措施。

不能，旧版本的 hdfs 仅支持节点间的数据平衡，新版本可通过 balancer 实现
1.手动重写所有数据 2.将数据全部移到几个节点上，再在节点间数据平衡

Q7 Spark 相较 MapReduce 的优势。

基于内存计算 RDD;基于 DAG 优化任务流程(延迟计算);易于部署,更低的框架开销;丰富的 API 支持。

Q8 描述高并发检索和综合搜索的场景特点，应用哪种技术来做支撑，指出数据和索引的存储位置。

高并发检索：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。技术：使用 hbase，数据可以存在 hbase 中，也可以存在 hdfs 中，索引表存在 hbase 中。
综合搜索：分布式实时文档存储，每个字段都可以被索引与搜索，分布式实时分析搜索引擎，支持高扩展，支持结构化和非结构数据。技术：使用 Elasticsearch， 数据和索引都存储在 ES 中。

Q9 描述 HDFS 的高可用性实现机制。

1.支持 NameNode HA。集群可以运行两个冗余的 NameNode，一个作为 active 节点提供服务，一个 standby 节点作为热备份，集群只能有一个 activeNameNode 为了保证 standby 节点与 active 节点之间数据的同步，两个节点会与另一组服务集群“journalNodes” (JNs)进行通信。2.利用 2N+1 台 JournalNode 存储 EditLog 3.最多容忍 N 台服务器挂掉 4.基于 Paxos 的一致性算法算法 5.额外一个 ZooKeeper 集群 6.每个 NameNode 上运行 ZKFailoverController (ZKFC)服务

Q10 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)一资源利用率低，无法交叉运行任务。一灵活性差。
Capacity Scheduler（容量调度器）:(思想)提前做预算，在预算指导下分享集群资源。(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。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。访问控制。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）。

Q11 场景优化思路

1.假设在 Inceptor 上执行任务，发现 Map Task 数量多、执行时间短，应采取哪种措施来提升性能?
对数据块进行合并：Automerge（碎片自动合并）
2.请简述在 Inceptor 中大表与大表做 join、大表与小表做 join 时分别有哪些优化手段。
大表与大表的普通 JOIN：实现普通 JOIN 的过程是这样的：扫描过滤两张表的数据（Map Stages），然后通过 Shuffle 将 Key 哈希值相同的数据分发到各个节点，在各节点内部执行 JOIN（Reduce Stages）
MapJoin 是一种针对大表与小表 JOIN 的特殊实现方式，在大小表数据量悬殊的情况下能有效的提升 JOIN 执行效率，一般受优化开关或者 Hint 控制启动。

HDFS：保证 namenode 高可用性不包括 datanode;namenode 安全模式错误的是允许用户读写;副本 3 大小 384M;正确是 AN 和 SN 不同机架;**hue** 实现是创建目录,上传文件,直接查看文件,更改权限;hue 支持 hive server2;hue 修改权限是 hdfs 相应权限,以 hdfs 用户登录;hdfs 高可靠是 journalNodes;namnode 数据形式是 fsimage 和 editlog
Hyperbase：23.33GB;全局索引正确倒排表,rowkey 一级索引,B+树检索;存储最小单元 region;不属于存储模型单位 region;正确的是都不正确;属于 hmaster 是分配 region,对 table 增删改查
Yarn：角色正确集群资源管理 rm,集群任务调度与管理;不包含 contianer;**RM** 错误是直接将资源分配;不正确是 nodemanager 负责调度 applicationMaster
inceptor：删除外表只删元数据不删数据文件，删除托管表都删;executor 正确是 fixed 和 ratio,配置的是逻辑 core 数量,1c:2G;无法看日志是 inceptorserver 节点/var/log/inceptorsql*/下;数据倾斜是 null 过多,union 合并;日志正确全选;与 hyperbase 正确是 inceptor 可访问 hyperbase,相辅相成
streamSQL：正确 stream 是数据流,application 是 steamjob 集合 **stream**：错误是转换规则变形
length 24 hour silde 1 minute。 index=department，dimension=sex,region 各部门 application 的 stramjob
zookeeper：不是功能存储大量数据;正确是选举机制确定 leader;通信 activeRM;分布式应用程序协调
flume：和 sqoop 错误是 flume 流式数据 sqoop 规范化数据,分布式处理数据;错误是功能相似可互替代;不支持 memory sink;不属于 source 是 filesource
sqoop：错误是 map 越多越好;抽取数据转换错误是设置 hdfs 文件列存储;**oozie** 调度正确 jdbc 正确上传,必须前提不存在;参数错误是 query 必须参数;oozie 正确是最简单,coordinator 包含;oozie 使用 ssh 是免密钥登录,有 bash 权限
elasticsearch：错误是数据存储在 hdfs;特性有误是都不正确;描述错误是只负责增删
Kafka：不是场景的数据迁移;不能保证可靠性是无法保证数据可靠 **Kerveros**：切换用户直接 kinit
Hadoop：场景正确 hive 数仓 hbase 检索 ES 全文 sparkstreaming 实时
spark：与 MR 比不是 spark 在 yarn 上运行 MR 不能 **mapreduce**：特点以上都是
LOAD：错误是源数据在 hdfs,通过 load 加载恢复
guardian 功能：用户管理、用户认证、审计、权限管理
各组件的**运维页面**描述不正确的是：通过 rm 的 8180 对 yarn 上运行的任务进行监控。
tableA10G 数据，优化效率：根据 id 字段分桶 有关 **minorCompact** 正确：把多个 HFile 合成一个。
不是**管理角色**：nodemanager。 不是**预安装**工作：配置安全模式。对**流处理计算框架**错误是都不对
电信部门 100 亿条：hyperbase+全局 **分桶表**通过改变数据存储分布对查询优化 ORC 事务表
不属于分布式计算框架 MATLAB wordcount 可能原因以上都有可能
maptask 数据合并参数：有 SET ngmr.partition **全局索引**语句：add_global_index
sqoop 查看关系数据库表：sqoop list-tables
flume sink 设置参数：org.apache.flume.sink.kafka.KafkaSink
开启 LDAP 连接 inceptor：beeline -u jdbc:hive2://$ip:10000 -n $username -p $password