大数据技术综述

**•大数据基本特征**

规模大、类型多、生处速度快、价值大密度低。

**•Hadoop经历了几个发展阶段，各有什么特点？**

三个，前hadoop从无到有；hadoop从有到成熟，后hadoop做了推广。3.0

TRA 
ECharts. D3. Cboard$ 
Impala, ArgoDB, Presto 
Hive, Inceptor, Spark SQL 
( & SQL SIX ) 
ElasticSerach, Search, Solr 
Spark Mllib, Sophon, Discover 
Storm, Flink DataStream, Spark Streaming, Slipstream 
MapReduce ( ) 
YARN, Mesos ( ) 
spark Core ( ) 
HBase, Hyperbase, Cassandra Redis Mongodb Ne04j, StellarDB ( 556itNo/New ) 
HDFS, Shiva ( ) 
Sqoop, Transporter 
Flume, Kafka **•大数据技术体系大致分为几层？每层包含哪些技术？**

**•ApacheHadoop项目包含哪些子项目？简述一下它们的功能。**

从核心讲有三项，1，星环一站式大数据平台 Transwarp Data Hub；2，星环人工智能 平台 Transwarp Sophon；3，星环数据云平台 Transwarp Data Cloud。

**•Spark包含哪些组件？简述一下它们的功能。**

Spark是一个计算的生态，spark的circle，swimming（四个）。

**•HDFS架构哪几种角色？各自承担什么功能？**

ActiveNameNode（AN）:管理命名空间、元数据、block副本策略；StandbyNameNode（SN）：AN宕机，快速升级为新的Active，同步元数据；NameNode元数据文件fsimage+edits；DataNode：存储Block和数据校验和、定期汇报状态。

**•为什么HDFS不合适存储大量的小文件？**

小文件过多，过多占用namenode内存，浪费block。文件过小，寻道时间大于数据读写时间，不符HDFS的设计。

**•Block副本的放置策略是什么？**

副本1：放Client所在节点；副本2：放不同的机架节点上；副本3：放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：优先空闲节点

**•HDFS离开安全模式的条件是什么？（只读）**

**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。

**•HDFS是如何实现高可用的？**

ActiveNN与StandbyNN的**主备切换**

利用**QJM（法定人数-奇2n+1）**实现元数据高可用

利用**ZooKeeper**实现Active节点选举

YARN

**•简述YARN与MapReduce的关系。**

MapReduce计算框架+资源管理系统；JobTracker既做资源管理，又做任务调度；YARN将JobTracker的资源管理、任务调度功能分离；

**•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。

**•Zookeeper在YARN中承担了哪些功能？**

Active节点选举；恢复ActiveRM的原有状态信息。

•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。

**•队列在资源调度中起什么作用？**

Client 
Master Node 
Manager 
Container 
ApplicationMaster 
ResourceManager 
Client 
NodeManager 
Container 
ApplicationMaster 
NodeManager 
Slave Node 
Container 
NodeManager 
Container 
NodeManager 
Container Container 
NodeManager 
Container 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。

**•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。

**•容量调度器会严格按预设比例分配资源吗？**

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。

**•简述公平调度器中队列权重和资源抢占的含义**。

**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。

**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。

分布式计算框架

**•简述MRSplit与HDFSBlock的关系。**

两者大小一样，但split是逻辑切片，block是物理切块。

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。两者大小一样，但split是逻辑切片，block是物理切块。

**•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。

**•简述Shuffle的工作原理。（必考）**

**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“key

hashcode%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。

**•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。

**•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**（弹性）

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。

**•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。

**•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。

**•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。

**•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。

分布式消息队列Kafka

**•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。

**•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。

**•PartitionLeader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，Leader负责读写，其他副本作为Follower从Leader同步消息。

**•为什么Zookeeper不亲自负责PartitionLeader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。

分布式SQL引擎 Inceptor

**• 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；基于Hadoop的数据仓库产品；分布式通用SQL引擎；基于Hive和Spark技术打造。 **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。

**• 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。

**• 分区目的？有几种类型？如何数据导入分区表？**

**目的：**减少不必要的全表扫描，提升查询效率。

**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，动态分区再静态分区建完之后；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只支持TEXT表、ORC表、CSV表和Holodesk表的分区操作。

**• 分桶目的？如何将数据导入分桶表？（必考）**

**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。--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：**• 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。

实时流处理引擎 Slipstream（事件驱动+微批处理）

**• 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.**微批将Input Stream按时间划分若干小数据块batch，事件驱动以单条数据被Input Stream接收为事件，逐条读取并处理。

**• 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。

**• 简述一下StreamJob的主要作用。**

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。

**• 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。

分布式搜索引擎 Search

**• Search数据模型与关系数据库的对应关系？**

**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）。

**• Seach包含哪几类节点，各自负责什么工作？**

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。

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。//集群规模大时有用，协调主节点和数据节点，据集群状态直接路由请求。

**• 简述Index、Document、Shard与副本Shard的关系。**

Search以Index为单位组织数据（Document），Type是Index的逻辑分类。Document是Search的最基础数据单元，以JSON格式存储。//**Shard**是Search的数据存储单元，是数据的容器，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”（Shard编号 = hash(Document id) % Shard数 量）的方式分配到不同的Shard中。Shard不可分割，但可以收缩（Shrink）。

**分区副本:**Shard分为主Shard和副本Shard，后者是前者的精确复制，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。

**• 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。

分布式NewSQL数据库 Hyperbase

**• 为什么将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。

• **为什么Hyperbase是一个Key-Value数据库？**

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。

**• 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：**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。//**Store：**一个Region由多个Store组成，每个Store存储一个列族。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。

**• 为什么要进行Region Split和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。

**• 简述HBase BulkLoad的基本过程。**

1.抽取：从数据源中抽取数据

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。

**简单题**

**1.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**。

**2.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。

**3.THD的4大组件的特性和使用场景。**

Inceptor

Hyperbase

StreamSQL

Discover

**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。