**大数据系统开发考试内容**

1. 大数据基本概念
2. 大数据特征

4V特征：体量Volume（**非结构化数据**的超大规模和增长）、多样性Variety（不同形式如文本、图像、视频、机器数据）、价值密度Value（大量的不相关信息）、速度Velocity（**实时分析**而非批量式分析）。

1. 大数据面临的挑战

数据量的指数级增加，做到低密度、高价值

1. 关系数据库与NoSQL数据库的差异

NoSQL数据库的主要分类：

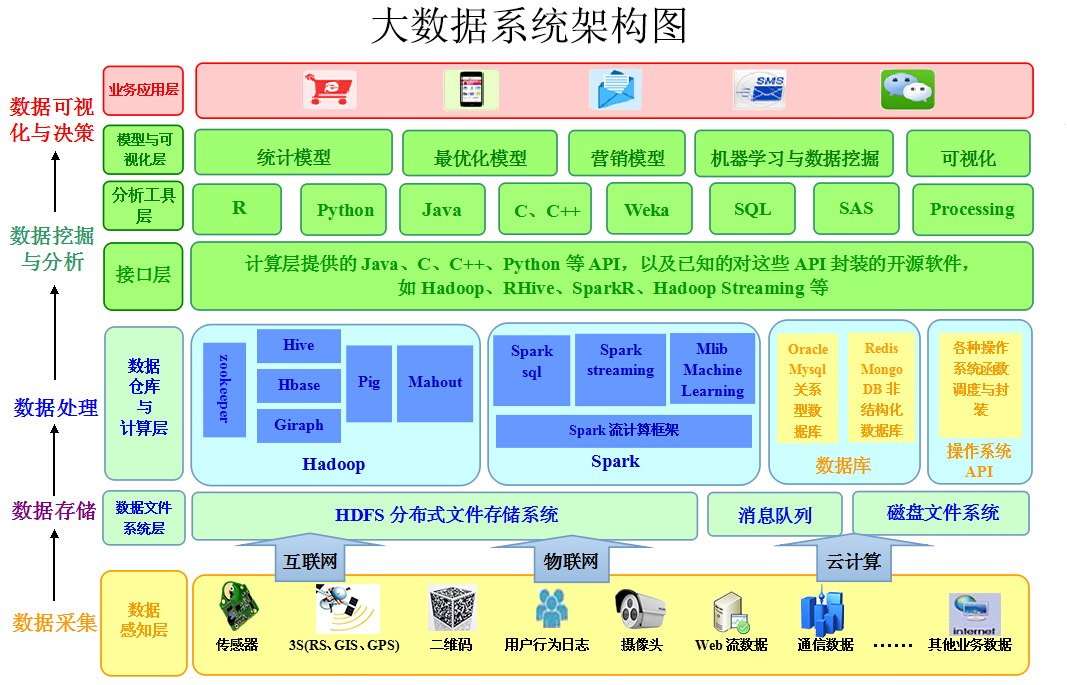
1. Key/Value存储——查询速度快
2. 列式存储——查找速度快，更容易进行分布式扩展
3. 文档存储——数据结构要求不严格、表结构可变
4. 图结构存储——高效匹配图结构相关算法。

与关系数据库的差异：

1. 存储方式：关系数据库——表格式，行/列；NoSQL——键值对、文档、图结构
2. 存储规范：关系数据库——规范性，避免冗余；NoSQL——鼓励冗余
3. 存储扩展：关系数据库——主要是纵向扩展（scale up，加cpu主内存等资源）；NoSQL——横向扩展（scale out）
4. 查询方式：关系数据库——结构化查询；NoSQL——非结构化查询语言SQL
5. 存储方式：关系数据库——主要在磁盘中；NoSQL——主要在内存中
6. 并发支持：关系数据库——通过事务和锁来支持并发，高并发情况下，执行效率较低；NoSQL——打破关系数据库范式的约束和事务一致性，并发性能高。
7. 大数据库关键技术
8. 分析技术：
   1. 数据处理：自然语言处理
   2. 统计与分析：A/B test；top N排行榜
   3. 数据挖掘：关联分析；分类；聚类
   4. 模型预测：预测模型；机器学习
9. 大数据技术：
   1. 数据采集：ETL（extract抽取-transform转换-load加载）工具
   2. 数据存取：关系数据库；NoSQL
   3. 基础架构支持：云存储；分布式文件系统
   4. 计算结果展现：云计算；标签云；关系图
10. 存储：
    1. 结构化数据：查询、统计、更新等操作效率低
    2. 非结构化数据（图片、视频、word、pdf等文件存储）：不利于查询、检索
    3. 半结构化数据：转换为结构化存储；按照非结构化存储
11. 解决方案：
    1. Hadoop（MapReduce技术）
    2. 流计算
12. 大数据处理流程

数据采集——数据存储——数据管理（数据处理、数据挖掘与分析、数据可视化与决策）

1. 大数据系统架构



1. Hadoop
2. 什么是Hadoop

Hadoop是一个能够对大量数据进行分布式处理的软件框架；是一个能够让用户轻松架构和使用的分布式计算平台；是一个能够开发和运行处理海量数据的应用程序。

1. Hadoop的核心架构

HDFS（为海量数据提供了存储）+MapReduce（为海量数据提供了计算框架）



1. Hadoop生态圈

图示

描述已自动生成

Hadoop核心组件包括：

分布式文件系统HDFS

分布式数据库存储系统HBase

分布式计算构架MapReduce

1. Hadoop的优点
2. 高可靠性。Hadoop按位存储和处理数据的能力值得人们信赖
3. 高扩展性。Hadoop是在可用的计算机集簇间分配数据并完成计算任务的，这些集簇可以方便地扩展到数以千计的节点中。
4. 高效性。Hadoop能够在节点之间动态地移动数据，并保证各个节点的动态平衡，因此处理速度非常快。
5. 高容错性。Hadoop能够自动保存数据的多个副本，并且能够自动将失败的人物重新分配。
6. 低成本。Hadoop是开源的，项目的软件成本大大降低。
7. Hadoop处理大数据的优势
8. HDFS（<https://zhuanlan.zhihu.com/p/158878632>）：被设计成适合运行在通用硬件上的分布式文件系统，java编写的；HDFS核心数据结构是Editlog+FsImage
9. HDFS的目标
10. 硬件故障

是常态，不是异常。整个HDFS系统将由数百或数千个存储着文件数据片段的服务器组成。HDFS里总是有一些部件是失效的。因此，故障的检测和自动快速恢复是一个很核心的设计目标。

1. 流式数据访问

运行在HDFS之上的应用程序必须流式地访问数据集。HDFS被设计成适合批量处理的，而不是用户交互式的。重点在于数据吞吐量。去掉POSIX（可移植操作系统接口）一小部分关键语义可以获得更好的数据吞吐率。

1. 大数据集

HDFS文件大小GB/TB级别。所以HDFS应该提供很高的聚合数据带宽，一个集群中支持数百个节点，还应支持千万级别的文件。

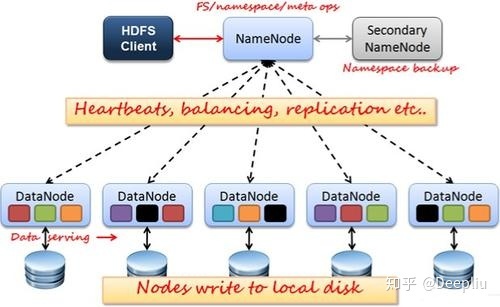
1. 简单一致性模型

大部分HDFS程序对文件操作都是一次写多次读的操作模式。一个文件一旦创建、写入、关闭之后就不需要修改了。这个简单化了数据一致的问题，并使高吞吐量的数据访问变得可能。

1. 移动计算比移动数据更经济
   1. 在靠近计算数据所存储的位置进行计算，消除网络拥堵，提高系统的整体吞吐量。
2. 异构软硬件平台间的可移植性

HDFS被设计成可以简便地实现平台间的迁移。

1. HDFS体系结构



名字节点（NameNode）和数据节点（DataNode）

图示, 示意图

描述已自动生成

Namenode、client、datanode通信协议：建立在TCP/IP之上。

Namenode——client之间的协议：Client Protocol；

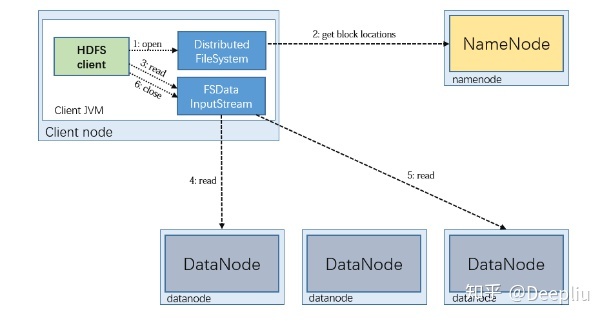
Namenode——datanode之间的协议：Datanode Portocol。

上述二者被封装在RPC（Remote Procedure Call）中。Namenode不主动发起RPC，它只被动地对datanode和namenode发起的RPC作出反馈。

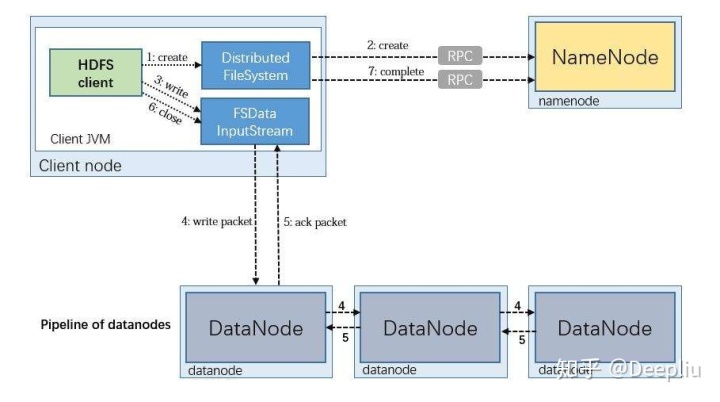
1. HDFS块副本（block Relication）
   1. 每一个文件可以配置块副本数量，默认是3。Datanode向namenode发送的块报告包含数据节点所拥有的数据块的列表。
   2. 副本的作用：防止因DataNode故障而导致数据丢失，除此之外块副本还可以提高块可读取的节点。
   3. 文件块的副本数量配置作为块元数据的一部分保存在NameNode中。
   4. HDFS默认的副本放置策略是第一个副本放在本地节点，第二个放在本地机架（rack）上的另外一个节点，第三个放在不同机架的节点上。
   5. 副本选择：就近，同机架优先，本地数据中心的副本优先。
   6. 安全模式：
      1. 启动时进入安全模式，不允许发生块复制。Namenode接收来自datanode的heartbeat和块报告。
      2. 每个块又一个特定的最小复制数。当namenode检查一个块大于最小复制数则认为已经安全复制。当达到配置的块安全复制比例时（加上额外的30秒），namenode就退出安全模式。并将小于特定复制数的块复制到其他的数据节点。
   7. 负载重平衡（Cluster Rebalancing）：在datanode的可用空间处于阈值以下时将自动将数据转移到其他datanode；当一个文件遇到高存取需求时，动态创建数据副本（还没实现）
2. HDFS元数据：HDFS的命名空间由namenode来存储。
   1. 元数据分别存在磁盘和内存中，磁盘维护数据的持久化，内存提高读写性能，两个地方的数据是一样的。元数据包含三个部分：目录书结构；文件与块的映射关系；datanode中块的存储信息
   2. EditLog事务日志：持久记录每一个对文件系统元数据的改变（如在HDFS中创建一个新的文件）；EditLog在本地文件系统中用一个文件存储。
   3. FsImage：整个文件的系统命名空间，包括文件块的映射表和文件系统的配置；存放在namenode的本地文件系统中。Namenode在内存中保留一个完整的文件系统命名空间和文件块映射表的镜像。
   4. 检查点：Namenode启动时，从磁盘中读取FsImage和EditLog，将EditLog中的所有事务应用到FsImage的仿内存空间，然后将新的FsImage刷新到本地磁盘。当前实现中，检查点仅在namenode启动的时候发生。
   5. 当datanode启动时，它将扫描它的本地文件系统，根据本地的文件产生一个所有HDFS数据库块的列表并报告给namenode（块报告）
3. HDFS读写流程（<https://blog.csdn.net/m0_48379126/article/details/120199221>）

Distributed FileSystem是一个类，FSData InputStream是一个类。

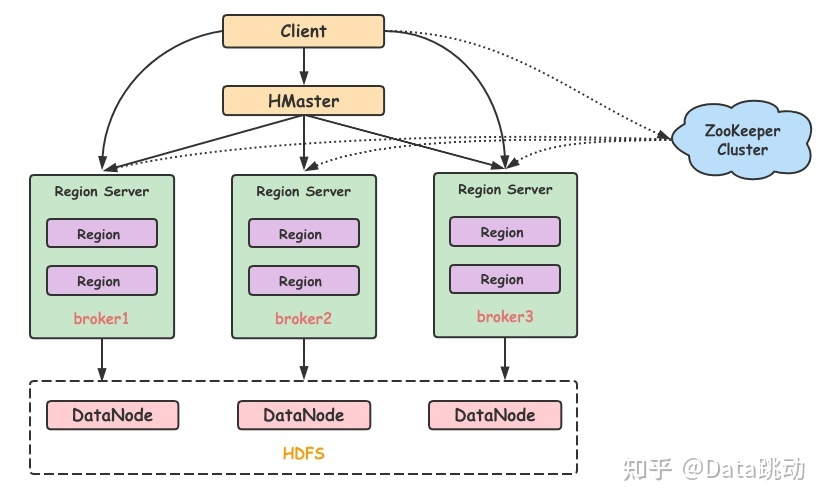
1. 读：client通过DistributedFileSystem对象的open()打开希望读取的文件；client对FSDataInputStream调用read()读取块数据；调用close()

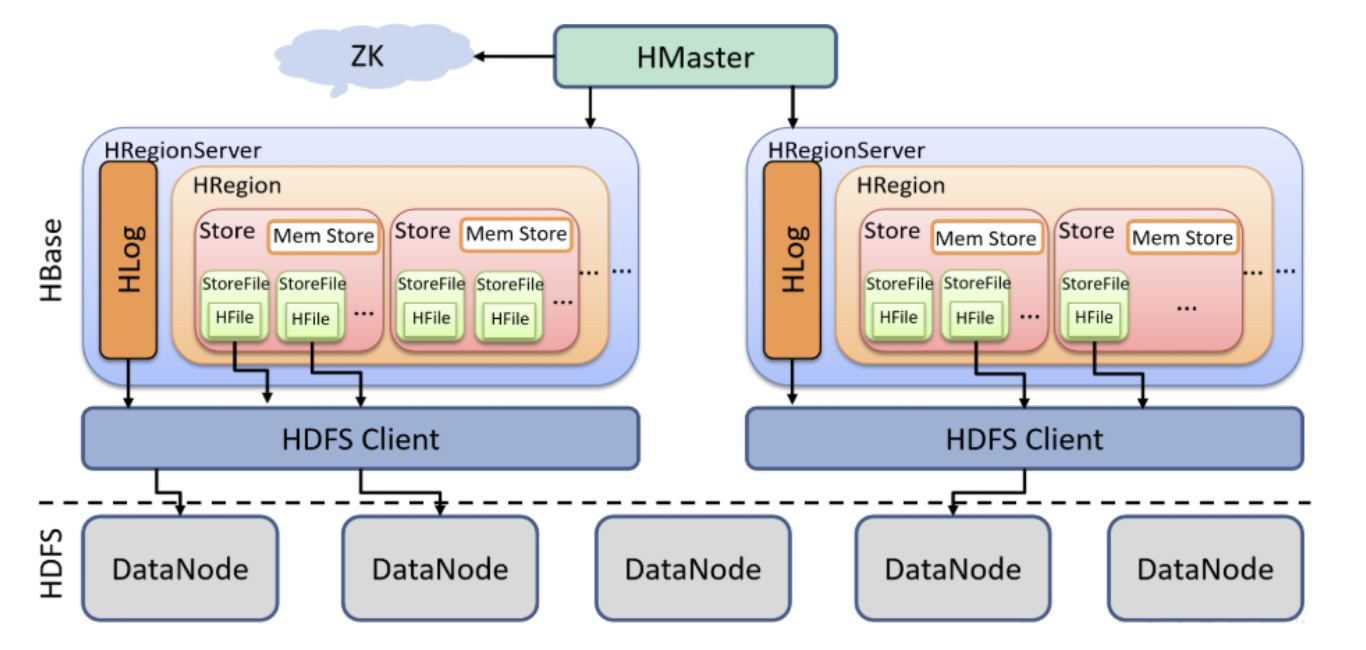


1. 写：调用create()；写入时，有两个队列，一个是“数据队列”，用于存放要写入的packet；一个是“确认队列”，用于接收datanode发来的确认Ack。



1. HDFS访问方式（Accessibility）
   1. DFSShell（重点）：用户可以和HDFS中的数据交互。命令集的语法类似bash
   2. DFSAdmin命令集是用于管理dfs集群的，这些命令只由HDFS管理员使用。
   3. 浏览器接口：用户可以通过web浏览器查看HDFS命名空间的内容。
   4. Java api
2. 数据组织：
   1. 数据块
      1. block：最大的一个单位，最终存储于datanode上的数据粒度；一般大小64/128MB
      2. packet：中等的单位，数据由DFSClient流向datanode的数据粒度，一般大小64KB
      3. chunk：最小的单位，DFSClient到datanode数据传输中校验的力度，一般大小512B
   2. 阶段状态：client创建一个文件的请求不会一开始就立即转发给namenode，先将文件数据缓存在本地的临时文件中；当本地文件堆积到一个HDFS块大小的时候，客户端才会通知namenode；namenode将文件名插入到文件系统层中，为它分配一个数据块，namenode构造包括数据节点ID和目标数据块标识的报文，以回复client的请求；client收到后将本地的临时文件刷新到指定的datanode数据块中。
3. HBase：一个面向列式存储的分布式数据库
4. HBase的特点
   1. 易扩展：一方面机遇运算能力（RegionServer）的扩展，通过增加RegionServer节点数量提升HBase上层的处理能力；一方面基于存储能力的扩展（HDFS），增加datanode节点数量。
   2. 海量存储：HBase是一个开源的分布式Key-Value数据库
   3. 高可靠性：WAL(Write Ahead Log)机制保证数据写入时不会因集群异常而导致写入数据丢失；Replication机制保证数据不会丢失或损坏。
   4. 稀疏性：在HBase的列族中，可以指定任意多的列，为空的列不占用存储空间，表可以非常稀疏。
5. HBase处理大数据
6. HBase体系结构





HMaster：负责管理RegionServer，实现其负载均衡；管理和分配Region；管理namespace和table的元数据（实际存储在HDFS上）；

RegionSever存放和管理本地Region，读写HDFS，管理Table中的数据， Client从HMaster中获取元数据，找到RowKey所在的RegionSever进行读写数据；

Zookeeper存放整个HBase集群的元数据以及集群的状态信息，（个人理解主要负责通信的工作），实现HMaster主从节点的failover。

1. HBase数据模型



1. 一条数据拥有一个全局唯一的键（RowKey）和任意数量的列（Column），一列或多列组成一个列族，HBase将表按照主键划分为多个Region存储在不同RegionServer上。
2. Column Family，HBase基s于列划分数据的物理存储。创建表的时候就要确定列组，一般数量<=3。
3. 访问HBase数据的方式有三种：1.基于RowKey的单行查询；2.基于RowKey的范围查询；3.全表扫描查询。
4. 相同RowKey的数据按照TimeStamp倒序排列，默认查询最新的版本。
5. 行式存储与列式存储的区别

行式存储：一行所有的信息放在一起。优点：insert/update容易；缺点：选择时可能只选择某几列但所有数据都会被读取。

列式存储：一列所有的信息放在一起。优点：投影很高效，任何列都能作为索引；缺点：insert/update比较麻烦。

表格

中度可信度描述已自动生成

1. HBase日志

WAL机制

1. HBase读写流程
   1. 写流程（不经过HMaster）：

图示

描述已自动生成

* + 1. 客户端首先访问zookeeper,获取存储元数据meta表所在的ReginServer地址.  
       2.客户端去访问该rs上的元数据,获取存储表的region分区所在的ReginServer地址.  
       3.客户端到对应rs从节点,将数据存储到该节点,先存在memstore内。对于client 来说,只需要将数据写入到memstore中就可以了,这也是Hbase快的原因  
       4.habse 会先将对数据的读写等操作记录到wal日志当中,然后才将数据写入进memstore中,(此时操作写入日志和数据写入memstore是事务性操作,要成功都成功,如果有一个失败,就会回滚)  
       WAL日志的目的就是为了数据安全  
       5.存在内存当中的memstore,会被手动或者自动fulsh到HDFS当中  
       6.fulsh后的hfile都是小文件,需要经过**Minor Compaction：小合并**,简单的将多个文件合成一个hfile,然后经过 **Major Compaction 大合并**将所有的文件合成一个文件,合并过程会清理过期,和删除的数据,消耗大量资源  
       7.当一个region内的数据量特别大的时候,如果在存入数据就会发生 split分裂,分裂成两个region。
  1. 读流程

图示

描述已自动生成

1.客户端访问zk集群,获取存储meta table所在的region server地址  
2.将meta table的内容缓存到本地,并设置有效期. 当第二次查询时,直接从本地读取meta table  
3.客户端到对应的rs上访问元数据,根据表名获取对应的region,根据每个region分区的rowkey范围 和rowley,获取要操作的 rs地址信息  
4.到对应rs上,读取memstore的数据；如果memstore上没有,就去BlockCache上读取,如果blockcache也没有就去storefile上读取,读取后将其缓存到BlockCache  
6.如果一直缓存会导致 BlockCache会越来越大,Hbase使用LRU(least recently used最近最久未使用)算法来清理缓存

1. MapReduce
2. MapReduce是什么

Hadoop Mapreduce是一个软件框架，基于该框架能够容易地编写应用程序，这些应用程序能够运行在由上千个商用机器组成的大集群上，并以一种可靠的、具有容错能力的方式并行地处理上TB级别的海量数据集。

Mapreduce=input+split+map+shuffle+reduce+finalize

1. MapReduce体系架构与处理流程

图示

描述已自动生成

多个mapper会有多个worker去完成，并且有一个Master worker（等级和其他worker相同）会作为用户的代理来协调整个过程，如让某个worker去拿对应的数据等等。

工作原理：

图示

描述已自动生成

（1）Client 启动一个作业 Job 并向 JobTracker 提交一个 JobId，运行作业所需要的资源文件比如 jar 包，Client 计算后输入的分片信息等拷贝到 HDFS 上。分片信息决定了 map 任务的数量，并由 JobTracker 分配计算。  
（2）JobTracker 接收到作业任务后，将作业放置在作业队列中，作业调度器根据自己的调度算法调度到某个作业时会跟根据[分片](https://www.zhihu.com/search?q=%E5%88%86%E7%89%87&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra=%7B%22sourceType%22%3A%22article%22%2C%22sourceId%22%3A%2236253130%22%7D)信息，为每个分片数据创建一个 map 任务，并将 map 任务分配给 TaskTracker 执行。需要注意的是 map 任务并不是随意分配给 TaskTracker （数据本地化Data Local：分配给含有该map任务的数据块中的tasktracker中，只有map考虑reduce不考虑，reducetasktracker根据key的hashcode%reducer数量的值分配，一个reducetask对应一个partition/region即分区）。每个节点上可以运行 map 的数量是有限的，TaskTracker 根据主机核的数量和内存的大小有固定数量的 map 槽和 reduce 槽。  
(3) TaskTracker 会定时的向 JobTracker 发送心跳，若一定时间没有收到心跳，JobTracker 就默认这个 TaskTracker 节点挂了要做资源回收，并把这个节点上的 task 重新分配到其他节点上。JobTracker 会监听 task 的执行情况，执行完成，JobTracker 会更新 Job 的状态为完成，若是一定数量的 task 执行失败，这个 Job 就会被标记为失败，JobTracker 将发送 Job 的运行状态信息给 Client 端。

处理流程：

图示

描述已自动生成

图示

描述已自动生成

一个maptask的环形缓存多次溢出，按分区合并文件，相同key值在同一个分区中；对分区中的key进行排序，相同key值的value放入同一集合中。对多个map任务的出书，按照不同的分区，网络copy到不同的reduce节点；对多个map的输出进行合并、排序；将最终的输出存入HDFS中。

1. MapReduce分区

图示

描述已自动生成

默认的Partitioner是HashPartitioner（继承自Partitioner）；

自定义分区重载getPartition函数即可。

1. MapReduce如何在分布式环境中处理词频统计、倒排索引
2. 实验

注：ppt上没有总结过来的内容（不在考纲上的内容？）：

HDFS：

快照（snapshot）流式复制（replication pipelining）

HBase：

数据压缩（在不丢失有用信息的情况下缩减数据量以减少存储空间，eg.对一些信息编码，在表中记录其编号）

MapReduce：

shuffle分为map shuffle和reduce shuffle（map shuffle为写入环形缓存，溢出进磁盘，进行分区文件归并；reduce shuffle为从不同块取出对应的数据，归并数据）其中combine（如果有）紧接在map操作后溢出前，作用为：减少写入磁盘的数据量；减少网络copy的数据量

Hadoop的“灵魂所在”：map、reduce端溢写时反复的排序、合并。