1. 简述HDFS的可靠性是如何设计的？

1. 安全模式 ：HDFS 刚刚启动时，NameNode 会进入安全模式（safe mode）。处于安全模式的NameNode不能做任何的文件操作，甚至内部的副本创建也是不允许的。NameNode 此时需要和各个DataNode 通信，获得DataNode 保存的数据块信息，并对数据块信息进行检查。只有通过了NameNode 的检查，一个数据块才被认为是安全的。当认为安全的数据块所占的比例达到了某个阈值（可配置），NameNode 才会退出。   
2. SecondaryNameNode ：Hadoop 中使用SecondaryNameNode 来备份NameNode 的元数据，以便在NameNode 失效时能从SecondaryNameNode 恢复出NameNode 上的元数据。SecondaryNameNode 充当NameNode 的一个副本，它本身并不处理任何请求，因为处理这些请求都是NameNode 的责任。   
NameNode 中保存了整个文件系统的元数据，而SecondaryNameNode 的作用就是周期性（周期的长短也是可以配置的）保存NameNode 的元数据。这些元数据中包括文件镜像数据FsImage 和编辑日志数据EditLog。FsImage 相当于HDFS 的检查点，NameNode 启动时候会读取FsImage 的内容到内存，并将其与EditLog 日志中的所有修改信息合并生成新的FsImage；在NameNode 运行过程中，所有关于HDFS 的修改都将写入EditLog。这样，如果NameNode 失效，可以通过Secondary NameNode 中保存的FsImage 和EditLog 数据恢复出NameNode 最近的状态，尽量减少损失。   
3. 心跳包（HeartBeats）和副本重新创建（re-replication）   
在HDFS 运行过程中，一部分DataNode 因为崩溃或是掉线等原因，离开了HDFS 系统时，为了保证NameNode 和各个DataNode 的联系，HDFS 采用了心跳包（Heartbeat）机制。位于整个HDFS 核心的NameNode，通过周期性的活动来检查DataNode 的活性，就像跳动的心脏一样，所以，这里把这些包就叫做心跳包。NameNode 周期性向管理的各个DataNode 发送心跳包，而收到心跳包的DataNode 则需要回复。因为心跳包总是定时发送的，所以NameNode 就把要执行的命令也通过心跳包发送给DataNode，而DataNode收到心跳包，一方面回复NameNode，另一方面就开始了与用户或者应用的数据传输。   
如果侦测到了DataNode 失效，那么之前保存在这个DataNode 上的数据就变成不可用的。那么，如果有的副本存储在失效的DataNode 上，则需要重新创建这个副本，放到另外可用的地方。其他需要创建副本的情况包括数据块校验失败等。   
4. 数据一致性 ：一般来讲，DataNode 与应用数据交互的大部分情况都是通过网络进行的，而网络数据传输带来的一大问题就是数据是否能原样到达。为了保证数据的一致性， HDFS 采用了数据校验和（CheckSum）机制。创建文件时，HDFS 会为这个文件生成一个校验，校验和文件和文件本身保存在同一空间中。传输数据时会将数据与校验和一起传输，应用收到数据后可以进行校验，如果两个校验的结果不同，则文件肯定出错了，这个数据块就变成了无效的。如果判定数据无效，就需要从其他DataNode 上读取副本。   
5. 租约 ：在Linux 中，为了防止出现多个进程向同一个文件写数据的情况，采用了文件加锁的机制。而在HDFS 中，同样也需要一种机制来防止同一个文件被多个人写入数据。 这种机制就是租约（Lease）。每当写入文件之前，一个客户端必须要获得NameNode 发放的一个租约。NameNode 保证同一个文件只会发放一个允许写的租约，那么就可以有效防止出现多人写入的情况。   
6. 回滚 ：HDFS 与Hadoop 一样处于发展阶段。而某个升级可能会导致BUG 或者不兼容的问题，这些问题还可能导致现有的应用运行出错。这一问题可以通过回滚回到旧版本解决。HDFS 安装或者升级时，会将当前的版本信息保存起来，如果升级之后一段时间内运行正常，可以认为这次升级没有问题，重新保存版本信息，否则，根据保存的旧版本信息，将HDFS 恢复至之前的版本。

2. 简述Hadoop MapReduce架构v1.0和v2.0的主要区别。

1.从Hadoop整体框架来说

Hadoop1.0由分布式存储系统HDFS和分布式计算框架MapReduce组成，其中HDFS由一个NameNode和多个DateNode组成，MapReduce由一个JobTracker和多个TaskTracker组成。

Hadoop2.0为克服Hadoop1.0中的不足进行了下面改进：

(1)、针对Hadoop1.0单NameNode制约HDFS的扩展性问题，提出HDFS Federation，它让多个NameNode分管不同的目录进而实现访问隔离和横向扩展，同时彻底解决了NameNode单点故障问题；

(2)、针对Hadoop1.0中的MapReduce在扩展性和多框架支持等方面的不足，它将JobTracker中的资源管理和作业控制分开，分别由ResourceManager（负责所有应用程序的资源分配）和ApplicationMaster（负责管理一个应用程序）实现，即引入了资源管理框架Yarn。

(3)、Yarn作为Hadoop2.0中的资源管理系统，它是一个通用的资源管理模块，可为各类应用程序进行资源管理和调度，不仅限于MapReduce一种框架，也可以为其他框架使用，如Tez、Spark、Storm等

2.从MapReduce计算框架来讲

(1)、MapReduce1.0计算框架主要由三部分组成：编程模型、数据处理引擎和运行时环境。它的基本编程模型是将问题抽象成Map和Reduce两个阶段，其中Map阶段将输入的数据解析成key/value，迭代调用map()函数处理后，再以key/value的形式输出到本地目录，Reduce阶段将key相同的value进行规约处理，并将最终结果写到HDFS上；它的数据处理引擎由MapTask和ReduceTask组成，分别负责Map阶段逻辑和Reduce阶段的逻辑处理；它的运行时环境由一个JobTracker和若干个TaskTracker两类服务组成，其中JobTracker负责资源管理和所有作业的控制，TaskTracker负责接收来自JobTracker的命令并执行它。

(2)、MapReducer2.0具有与MRv1相同的编程模型和数据处理引擎，唯一不同的是运行时环境。MRv2是在MRv1基础上经加工之后，运行于资源管理框架Yarn之上的计算框架MapReduce。它的运行时环境不再由JobTracker和TaskTracker等服务组成，而是变为通用资源管理系统Yarn和作业控制进程ApplicationMaster，其中Yarn负责资源管理的调度而ApplicationMaster负责作业的管理。

3. 简述HBase的基本数据模型

HBase 不支持关系模型，它可以根据用户的需求提供更灵活和可扩展的表设计。与传统的关系型数据库类似，HBase 也是以表的方式组织数据，应用程序将数据存于 HBase 的表中，HBase 的表也由行和列组成。但有一点不同的是，HBase 有列族的概念，它将一列或多列组织在一起，HBase 的每个列必须属于某一个列族。

1.表（Table）:HBase 中的数据以表的形式存储。同一个表中的数据通常是相关的，使用表主要是可以把某些列组织起来一起访问。表名作为 HDFS 存储路径的一部分来使用，在 HDFS 中可以看到每个表名都作为独立的目录结构。

2.行（Row）:在 HBase 表里，每一行代表一个数据对象，每一行都以行键（Row Key）来进行唯一标识，行键可以是任意字符串。在 HBase 内部，行键是不可分割的字节数组，并且行键是按照字典排序由低到高存储在表中的。在 HBase 中可以针对行键建立索引，提高检索数据的速度。

3.列族（Colunm Family）:HBase 中的列族是一些列的集合，列族中所有列成员有着相同的前缀，列族的名字必须是可显示的字符串。列族支持动态扩展，用户可以很轻松地添加一个列族或列，无须预定义列的数量以及类型。所有列均以字符串形式存储，用户在使用时需要自行进行数据类型转换。

4.列标识（Column Qualifier）:列族中的数据通过列标识来进行定位，列标识也没有特定的数据类型，以二进制字节来存储。通常以 Column Family：Colunm Qualifier 来确定列族中的某列。

5.单元格（Cell）:每一个行键、列族、列标识共同确定一个单元格，单元格的内容没有特定的数据类型，以二进制字节来存储。每个单元格保存着同一份数据的多个版本，不同时间版本的数据按照时间先后顺序排序，最新的数据排在最前面。单元格可以用 <RowKey,Column Family: Column Qualifier,Timestamp> 元组来进行访问。

6.时间戳（Timestamp）:在默认情况下，每一个单元格插入数据时都会用时间戳来进行版本标识。读取单元格数据时，如果时间戳没有被指定，则默认返回最新的数据；写入新的单元格数据时，如果没有设置时间戳，默认使用当前时间。每一个列族的单元数据的版本数量都被 HBase 单独维护，默认情况下 HBase 保留 3 个版本数据。