第三章 分布式文件系统HDFS

第三章考点

- 分布式文件系统基本概念
- 块的定义、元数据的概念、主从节点的功能、名称节点的数据结构、第二名称节点的工作流程
- HDFS存储原理，数据存放策略，数据错误处理
- HDFS数据读写过程

一、分布式文件系统

1. 分布式文件系统（HDFS）把**文件分布存储到多个计算机节点**上，成千上万的计算机节点**构成计算机集群**
2. 计算机的集群结构

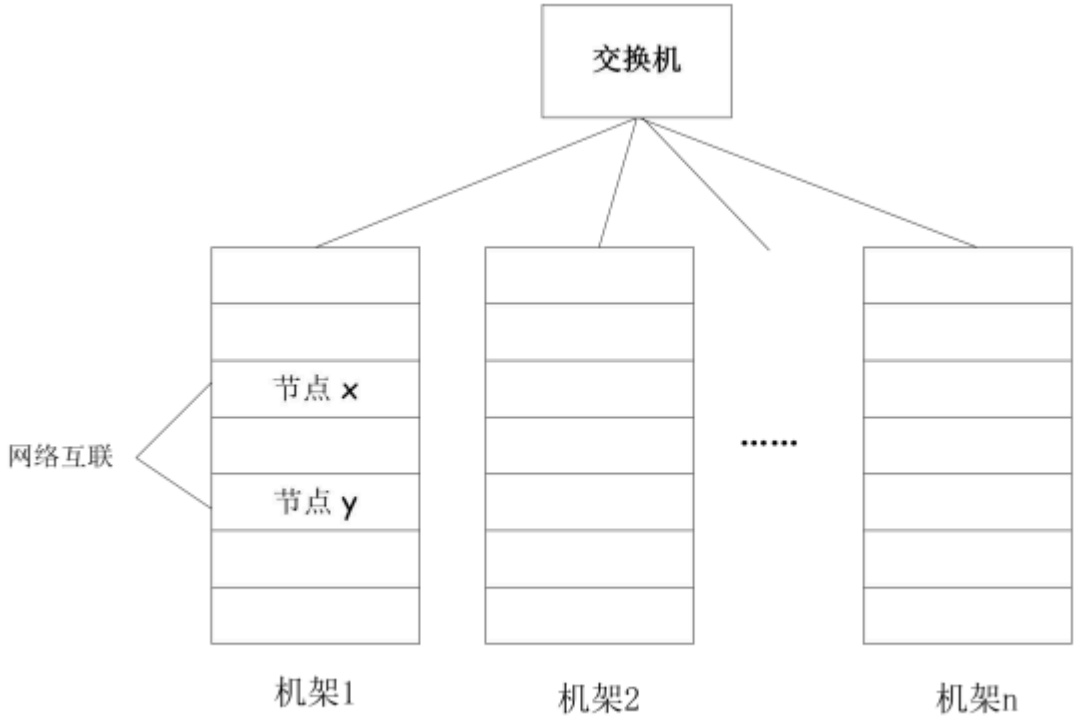


图3-1 计算机集群的基本架构

节点之间通过网络互连，机架之间用交换机互联

3. **分布式文件系统的结构**
   分布式文件系统在**物理结构上**是由计算机集群中的多个节点构成的，这些节点分为两类，

一类叫做主节点或者**名称节点**，另一类叫做从节点或者**数据节点**

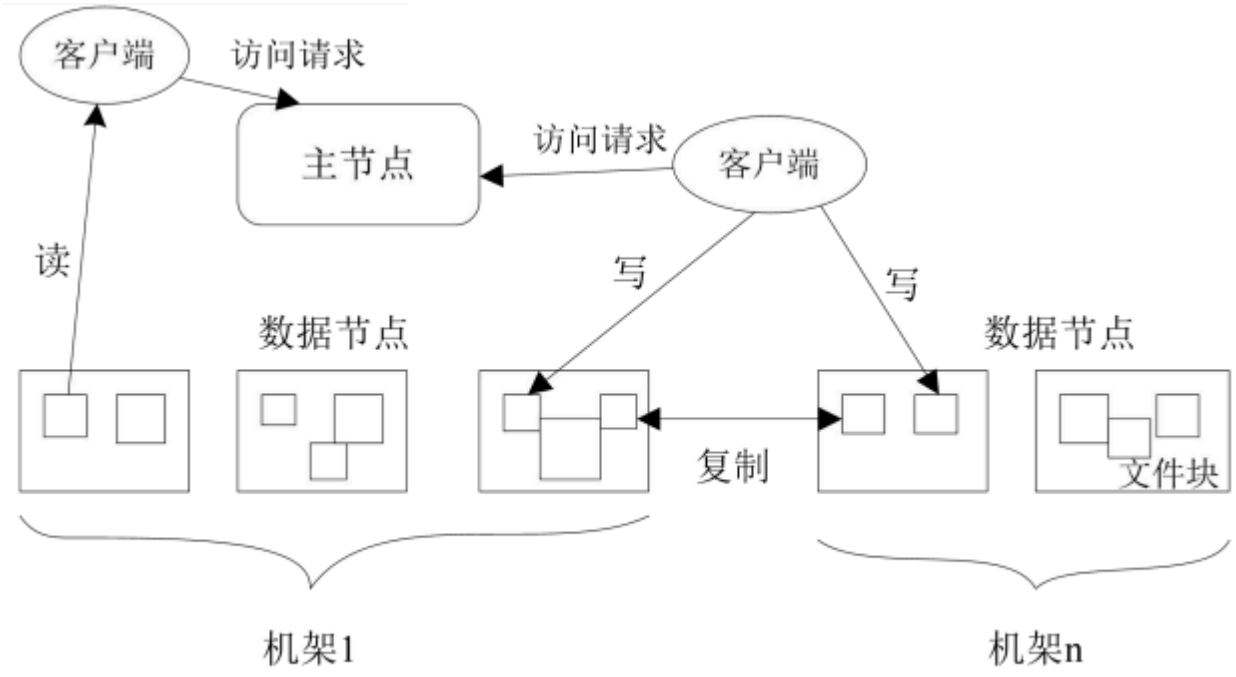


图3-2 大规模文件系统的整体结构

二、HDFS的简介及概念

1. HDFS要**实现以下目标**：
   - 兼容廉价的硬件设备：几十万的小型机→普通PC机
   - 数据流读写：批量数据处理，流式访问数据
   - 支持大数据集
   - 简单地文件模型：一次写入多次读取。无法修改文件，只能读取
   - 强大的跨平台兼容性：Java语言
2. **应用局限性**
   - 批量处理数据，较高延迟，不适合低延迟的数据访问
   - 无法高效存储大量小文件：检索效率低
   - 不支持多用户写入及任意修改文件
3. **块(Block)**

   块是HDFS中的最核心的概念，HDFS中默认一个块128MB，一个文件被分为多个块，以块作为存储单位

   块的大小远远大于普通文件系统，可以**最小化寻址开销**

   块的大小不宜过大，因为MapReduce中**一次只能处理一个块中的数据**，如果**启动任务太少，反而会降低作业并行处理速度**

   HDFS采用抽象的块概念可以带来以下几个明显的**好处**：
   - **支持大规模文件存储**：文件以块为单位进行存储，一个大规模文件可以被拆分成若干文件块，**不同的文件块可以被分发到不同的节点上**，因此，一个文件块的大小不会受到单个节点的存储容量的限制，可以远远大于网络中任意节点的存储容量
   - **简化系统设计**：首先，大大简化了存储管理，因为文件块大小是固定的，这样就可以**很容易计算出一个节点可以存储多少文件块**；其次，方便了元数据的管理，元数据不需要和文件块一起存储，可以由其他系统负责管理元数据

- **适合数据备份**：每个文件块都可以**冗余存储在多个节点**上，大大提高了系统的容错性和可用性

  元数据的概念：

  元数据描述文件是什么，文件被分为多少块，每个块和文件是怎么映射的，每个块被存储在哪个服务器上

4. 名称节点和数据节点

   主从节点的功能

	NameNode	DataNode
存储内容	存储元数据	存储文件内容
保存位置	元数据保存在内存中	文件内容保存在磁盘
保存的关系	保存文件，block，datanode之间的关系	维护了block id到datanode本地文件的映射关系

- **名称节点**

  名称节点主要以元数据的形式进行管理和存储，用于维护文件系统名称并管理客户端对文件的访问，**名称节点保存了两个核心的数据结构，即FsImage和EditLog**

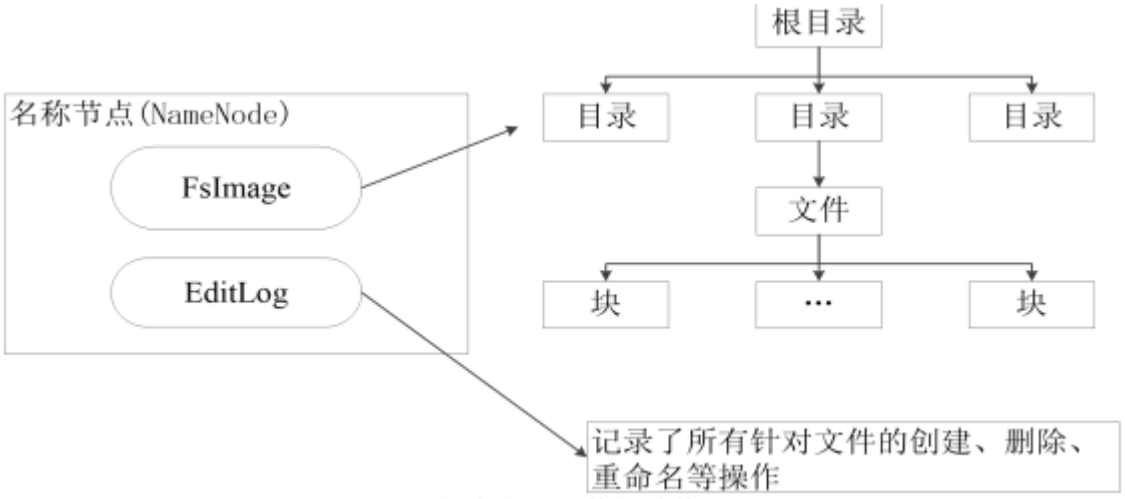


图3-3 名称节点的数据结构

其中FsImage用于**维护文件系统树以及文件树中所有文件和文件夹的元数据**

操作日志文件EditLog**记录了所有针对文件的创建、删除、重命名等操作** FsImage文件存储元数据，包含此类信息：**文件的复制等级、修改和访问时间、访问权限、块大小以及组成文件的块。对于目录，则存储修改时间，权限和配额元数据**

FsImage文件**没有记录每个块存储在哪个数据节点**，而是由名称节点把这些映射信息保存在内存中，当数据节点加入HDFS集群中时，**数据节点会把自己所包含的块列表告知给名称节点**，以后会定期执行这种告知操作，以确保名称节点的块映射是最新的

- **名称节点的启动过程**
  - 名称节点启动的时候，它会**将FsImage的内容加载到内存中**
  - **执行EditLog文件中的各项操作**，使得**内存中的元数据和实际同步，生成最新的元数据**，存在内存中的元数据支持客户端的读操作
  - **创建一个新的FsImage文件和一个空的EditLog文件**
- 第二名称节点

  随着名称节点中的Editlog的累积名称节点启动会变得非常慢

因此使用第二名称节点(SecondaryNameNode)
第二名称节点是HDFS框架中的一个组成部分，它是用来保存名称节点中**对HDFS元数据信息的备份，并减少名称节点重启的时间**。SecondaryNameNode一般是**单独运行在另一台机器上**

第二名称节点的工作流程：

- 第二名称节点会定期和名称节点进行通信，请求其**停止使用EditLog文件**，暂时将**新的操作写到一个新的文件edits.new**上来。这个操作是瞬间完成的
- 第二名称节点通过**HTTP GET的方式从名称节点中获取FsImage和EditLog文件，并且下载到本地相应目录下**
- 第二名称节点将**下载下来的FsImage 载入到内存**，然后一条一条地**执行EditLog文件中的各项更新操作**，使得**内存中的FsImage保持最新**；这个过程就是EditLog和FsImage文件合并；
- 第二名称节点执行完更新操作后，会通过**POST的方式将新的FsImage文件发送到NameNode节点上**
- NameNode将从SecondaryNameNode接受到的**新的FsImage替换旧的FsImage**同时将**edits.new替换EditLog文件**，通过这个过程EditLog就变小了

- **数据节点**

数据节点是分布式文件系统HDFS的工作节点，负责**数据的存储和读取**，会**根据客户端或者是名称节点的调度来进行数据的存储和检索**，并且**向名称节点定期发送自己所存储的块的列表**
每个数据节点的数据会被保存在各自节点的本地Linux文件系统中

三、HDFS体系结构

HDFS采用**主从结构**模型，一个HDFS集群包括**一个名称节点和若干数据节点**，名称节点只有一个，数据节点可以多个

1. 命名空间
   - HDFS的命名空间是指通过目录的嵌套关系来表示文件和目录之间的层级关系，所有的文件和目录都是从根目录开始的
   - HDFS采用的是传统的分级文件体系，因此用户可以像普通文件系统一样，创建删除目录和文件，在目录转移文件，重命名文件等
   - /目录名称
2. 通信协议
   HDFS是一个部署在集群上的分布式文件系统，因此，很多数据需要通过网络进行传输，所有HDFS通信协议都是构建在**TCP/IP协议**的基础之上
   客户端通过一个可配置的端口向**名称节点**主动**发起TCP连接**，并且使用客户端协议与名称节点进行交互
   名称节点和数据节点之间则使用**数据节点协议**进行交互
   客户端与数据节点之间的交互是通过远程过程调用RPC
3. **HDFS体系结构的局限性**
   HDFS1.0只设计了一个名称节点，这样做虽然大大简化了系统设计，但是也带来了一些明显的局限性

- 命名空间限制：名称节点是保存在内存中的，因此，名称节点能够容纳的对象(文件、块)的个数会受到内存空间大小的限制
- 性能的瓶颈：整个分布式文件系统的吞吐量受限于单个名称节点的吞吐量
- 隔离问题：由于集群中只有一个名称节点，只有一个命名空间，因此无法对不同的应用程序进行隔离
- 集群可用性：一旦这个唯一的名称节点发生故障，会导致整个集群变得不可用

四、HDFS存储原理

1. **冗余数据保存：**
   作为一个分布式文件系统，为了保证系统的**容错性和可用性**，HDFS采用多副本的方式对数据进行冗余存储，通常一个数据块的多个副本会被保存在不同的数据节点上
   冗余存储的优点：
   - 加快数据传输速度
   - 很容易检查数据错误
   - 保证数据可靠性
     数据存取规则：
   - HDFS默认冗余复制因子为3
   - 第一个副本放置在上传文件的数据节点，如果是在集群外提交的则随机挑选一个CPU不太忙的节点
   - 第二个副本放在与第一个副本不同机架的节点上
   - 第三个副本与第一个副本相同的其他节点上
   - 更多副本随机放置
2. 数据读取策略
   HDFS提供了一个**API**可以**确定一个数据节点所属机架的ID**，客户端也可以调用API来获取自己所属的机架
   当客户端读取数据时候，从名称节点获得数据块不同副本的位置列表，列表中包含了副本所在的数据节点，可以调用API来确定客户端和这些数据节点所属的机架ID，当发现**某个数据块副本对应的机架ID和客户端对应的机架ID相同时（距离最近）**，就优先选择该副本读取数据，如果没有发现，就随机选择一个副本读取数据
3. **数据错误与恢复**
   HDFS具有较高的容错性，可以兼容廉价的硬件，它把硬件出错看做一种常态，而不是异常，并且设计了相应的机制检测数据错误和进行自动恢复，主要包括以下几种情形：名称节点出错，数据节点出错和数据出错
   - **名称节点出错：**
     名称节点保存了所有的元数据信息，其中最核心的两大数据结构是FsImage和EditLog，如果这两个文件发生损坏，那么整个HDFS实例将失效
     因此，HDFS设置了备份机制，把这些核心文件同步复制到备份服务器SecondaryNameNode上，当名称节点出错时，就可以根据备份服务器SecondaryNameNode中FsImage和EditLog数据进行恢复
   - **数据节点出错：**
     每个数据节点会定期向名称节点发送状态信息，当数据节点发生故障，或者网络中

断，名称节点就无法收到来自数据节点的状态信息，这些数据节点就会被标记为”宕机“，节点上所有的数据都会被标记为不可读，名称节点不会再给它们发送任何输入输出请求

由于一些数据节点的不可用，可能会导致一些数据块的副本数量小于冗余因子，名称节点会定期检查这种情况，一旦发现某个数据块的副本数量小于冗余数据，就会启动冗余复制，产生新的副本

HDFS跟其他分布式文件系统的最大区别就是可以调整冗余数据的位置

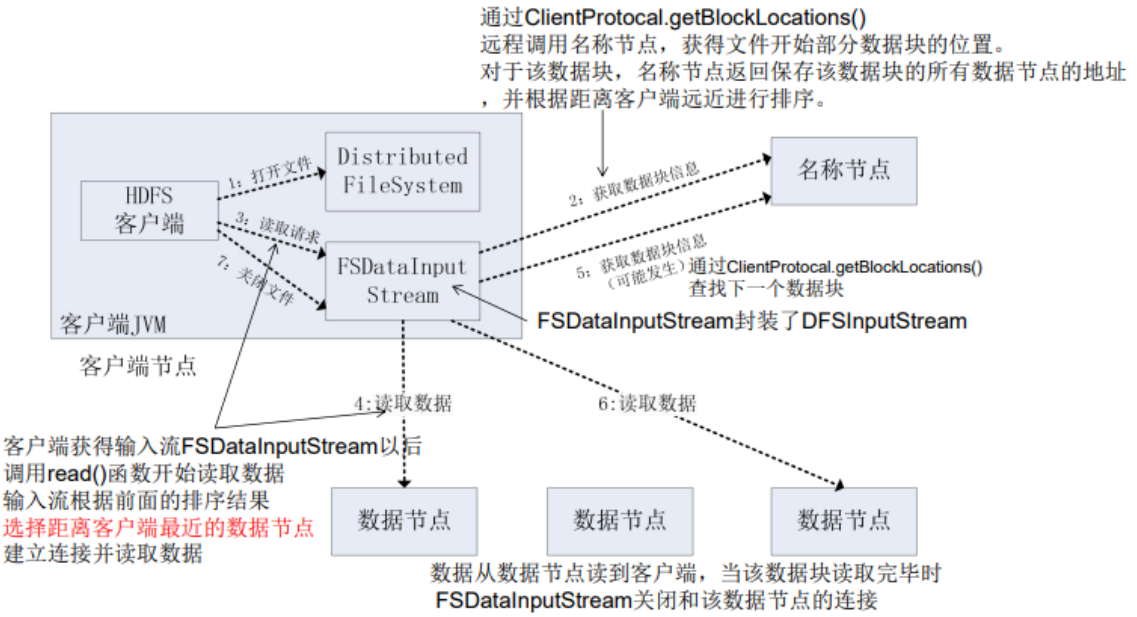
- **数据出错：**
  网络传输和磁盘错误等因素，都会造成数据错误
  客户端在读取到数据后，会采用校验码对数据块进行校验，以确定读取到正确的数据
  如果校验出错，客户端就会请求另外一个数据节点读取该文件块，并向名称节点报告这个文件块存在错误，名称节点会定期检查并且重新复制这个块

五、HDFS数据读写过程

1. HDFS相关类
   FileSystem是一个通用文件系统的抽象基类，可以被分布式文件系统继承

	FileSystem	DistributeFileSystem(HDFS中实现)
输入流 `open()`	FSDataInputStream	DFSInputStream
输出流 `create()`	FSDataOutputStream	DFSOutputStream

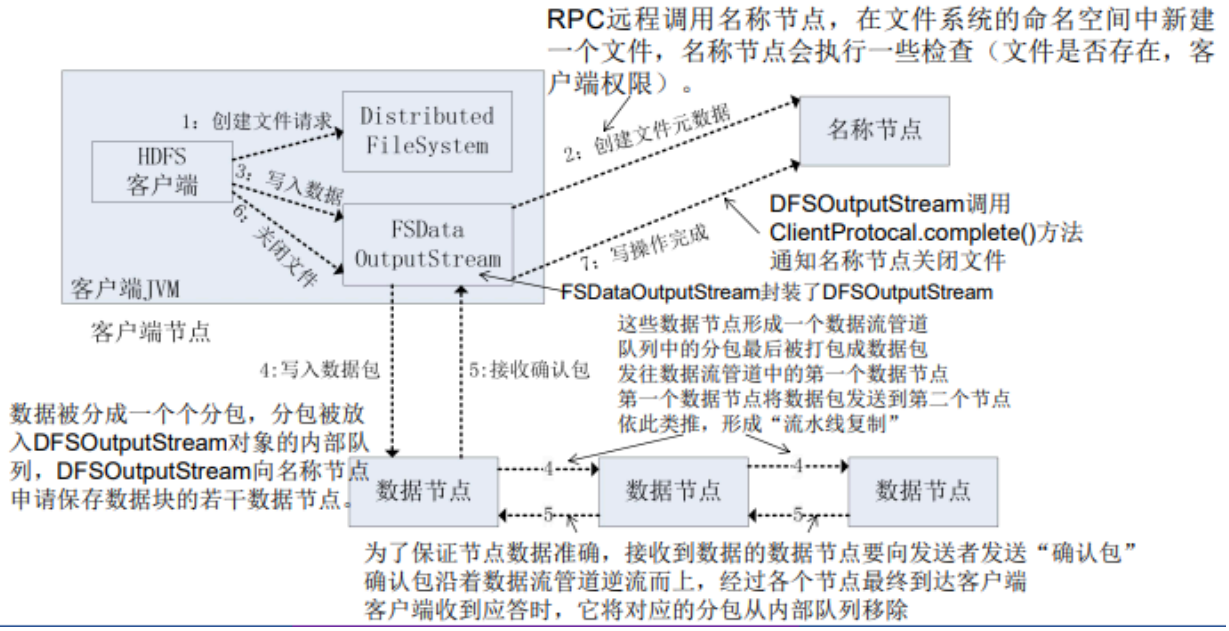
2. **==读数据过程 ==**



- 打开文件：从HDFS客户端打开文件分布式文件系统
- 获取数据块的信息：远程调用名称节点，获得文件开始部分的数据块的位置，对于该数据块名称节点返回并保存该数据块所有数据节点的地址，并根据距离客户端远近进行排序
- 发送读取请求：通过FSInputStream实现

- 读取数据：从数据节点读取数据
  客户端获得输入流FSDataInputStream以后，调用read()函数开始读取数据，输入如流
  根据前面的排序结果选择距离客户端最近的数据节点
  数据节点从数据节点读到客户端，当该数据块读取完毕时，FSDataInputStream关闭和该节点的连接
- 关闭文件

3. **写入数据**



- 由hdfs客户端创建文件请求发送到分布式文件系统
- 通过FSDataOutoutStream创建文件元数据，RPC远程调用名称节点，在文件系统命名空间中新建一个文件，名称节点会执行一些检查(文件是否存在，客户端权限)
- 通过写入数据节点
- 写入数据包，数据被分为一个个分包，分包被放入DFSOutputStream对象的内部队列，DFSOutputStream向名称节点申请保存数据块的若干数据节点
- 接受确认包，这些数据节点形成一个数据流管道，队列中的分包最后被打包成数据包，发往数据流管道中的第一个数据节点，第一个数据节点将数据包发送到第二个数据节点，形成流水线复制
  为了保证节点数据准确，接受数据的数据节点要向发送者发送”确认包“，确认包沿着数据流管道逆流而上，经过各个节点，最终到达客户端，客户端收到应答时，它将对应的分包从内部队列移除
- 数据写入完毕，关闭文件

六、HDFS实践

1. HDFS的shell命令
   Hadoop中有三种Shell命令方式
   hadoop fs适用于任何不同的文件系统，比如本地文件系统和HDFS文件系统
   hadoop dfs只能适用于HDFS文件系统
   hdfs dfs跟hadoop dfs的命令作用一样，也只能适用于HDFS文件系统
2. Hadoop所提出的解决方案

- 数据分布式存储，在HDFS中，数据被分为多个数据块(block)，这些数据块被分布在多个服务器上，这样，每个服务器只需要处理一部分数据，从而降低了单个服务器的负载。当数据量增长时，可以通过增加服务器数量来扩展系统的存储和处理能力
- 数据备份：为了防止数据丢失，HDFS会将每个数据块备份到多个服务器上。通常，一个数据块会有3个备份，分别存储在不同的位置，这样即使某个服务器出现问题，其他服务器上的备份数据仍然可以保证数据的完整性
- 容错机制：HDFS具有强大的容错机制，当某个服务器出现故障时，系统会自动检测并将故障服务器上的数据迁移到其他正常的服务器上，这样即使部分服务器出现问题，整个系统仍然可以正常运行
- 实时更新和查询：HDFS支持实时读写操作，可以满足公司对实时更新和查询的需求，同时HDFS还支持对大量数据进行并行处理，可以大大提高数据处理的效率
- 数据一致性：HDFS通过一种称为“一次写入，多次读取”的策略来保证数据的一致性，这意味着一旦数据被写入HDFS，就不会被修改。如果需要对数据进行更新，那么会创建一个新的数据块来存储更新后的数据，这样可以避免在并发访问时出现数据不一致的问题