（PPT1）

大家好，在本节中，我们将向同学们介绍HDFS的存储原理，包括数据的冗余存储、数据存取策略、数据的错误和恢复。这些都是分布式文件系统应该解决的问题，HDFS作为应用较为广泛的分布式文件系统，它又是如何解决这些问题的？

（PPT2）

首先我们来看看数据的冗余存储问题。为了保证系统的容错性和可用性，HDFS采用了多副本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分布到不同的数据节点上，如图1所示，数据块1被分别存放到数据节点A和C上，数据块2被存放在数据节点A和B上。这种多副本方式具有以下几个优点：

（1）加快数据传输速度：由于数据块被复制了多个副本存放到不同的数据节点上，当多个客户端访问某文件的同一个数据块的时候，就可以并行读取存储在不同数据节点的数据块，从而加快了数据的传输速度。

（2）容易检查数据错误：由于数据节点之间通过网络传输数据来复制数据块的，当数据传输出现错误，被复制的数据块就会出错，采用多副本可以很容易判断数据传输是否出错。

（3）保证数据可靠性：当某个数据节点的不可用时，会导致该数据节点上的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因子，就会启动数据冗余复制，为它生成新的副本，从而保证了数据的可靠性。

（PPT3）

将数据块多副本冗余存储到不同的数据节点上，HDFS集群上有那么多数据节点，名称节点NameNode选择哪些数据节点来存放数据块呢？冗余数据块有两种存放方式：

（1）同一机架上不同数据节点上

（2）不同机架上不同数据节点上

我们知道，一个HDFS集群有很多机架，不同机架之间数据通信需要经过路由器或交换机，而同一机架不同机器之间不需要，因此同一机架上机器间的通信比不同机架机器间的通信带宽大，如果仅仅考虑网络带宽我们选择第一种方式是可行的。

但是，机架发生故障是不可避免的，所有的副本放在一个机架上就丧失了数据的可靠性，因此第二种方式可以保证数据的可靠性。其次，副本冗余存储到不同机架上，可以在读数据的时候并发进行，大大提高了数据读取速度。

为了均衡数据的可靠性、可用性和网络带宽的利用率HDFS综合这两种方式。HDFS默认的冗余复制因子为3，每个数据块复制3份，

第一个副本：放在离客户端最近的机架上的一台磁盘不太满，CPU不太忙的数据节点上

第二个副本：放在与第一个副本不同的机架的节点上

第三个个副本：与第一个副本相同机架的不同数据节点上

这种策略减少了机架间的数据传输，提高了写操作的效率。机架的错误远远比节点的错误少，所以这种策略不会影响到数据的可靠性和可用性。与此同时，因为数据块只存放在两个不同的机架上，所以此策略减少了读取数据时需要的网络传输总带宽。在这种策略下，副本并不是均匀的分布在不同的机架上：三分之一的副本在一个节点上，三分之二的副本在一个机架上，其它副本均匀分布在剩下的机架中，这种策略在不损害数据可靠性和读取性能的情况下改进了写的性能。

（PPT4）

一个数据块有多个副本，客户端到底优先读取哪个DataNode节点上的该数据块的副本呢？这就是HDFS读取副本的选择策略，而这个工作具体是由NameNode来完成的。

NameNode采用就近策略，其基本原理就是按照客户端与DataNode节点之间的距离进行排序，该算法的基本思路如下：

1.如果该数据块的一个副本存在于客户端，则客户端优先从本地读取该数据块；

2.如果该数据块的一个副本与客户端在同一个机架上，且没有一个副本存放在客户端，则客户端优先读取这个同机架上的副本；否则客户端优先读取同机器的副本，失败的情况下然后再优先考虑这个同机架上的副本；

3.如果该数据块既没有一个副本存在客户端，又没有一个副本与客户端在同一个机架上，则随机选择一个DataNode节点作为优先节点。

由于HDFS提供了一个API可以确定数据节点和客户端所属的机架ID，因此客户端读取数据时，从名称节点获取数据块不同副本所在的数据节点信息，通过调用API获得这些数据节点和自身的所在机架ID号，就可以根据该算法来确定读取哪个数据节点上的副本了。

（PPT5）

数据块的多个副本是如何复制并存储到不同数据节点的？一种做法是客户端按照名称节点的要求复制副本，然后逐个上传到指定的数据节点上。这种做法效率较低，HDFS的数据复制采用了流水线复制的策略，大大提高了数据复制过程的效率。

其过程如下：

当客户端要往HDFS中写入一个文件时，这个文件会首先被写入本地，并被切分成若干个块，每个块的大小是由HDFS的设定值来决定的。

每个块都向HDFS集群中的名称节点发起写请求，名称节点会根据系统中各个数据节点的使用情况，选择一个数据节点列表返回给客户端，然后，客户端就把数据首先写入列表中的第一个数据节点，同时把列表传给第一个数据节点。

当第一个数据节点接收到4KB数据的时候，写入本地，并且向列表中的第二个数据节点发起连接请求，把自己已经接收到的4KB数据和列表传给第二个数据节点。

当第二个数据节点接收到4KB数据的时候，写入本地，并且向列表中的第三个数据节点发起连接请求，依此类推，列表中的多个数据节点形成一条数据复制的流水线。

最后，当文件写完的时候，数据复制也同时完成。