

# 分布式文件系统HDFS

---

## 1、HDFS实现目标

兼容廉价的硬件设备

实现流数据读写

支持大数据集

支持简单的文件模型

强大的跨平台兼容性

## 2、局限性

不适合低延迟数据访问

无法高效存储大量小文件

不支持多用户写入及任意修改文件

## 3、相关概念

### 块

- 为什么要这么设计？（块容量设置大，有界限）

支持面向大规模数据存储

降低分布式节点的寻址开销

- 抽象的块概念设计好处

支持大规模文件存储----将容量数据切成小块，这些小块可以分布地存储在不同机器上吗，突破单机存储容量的上限

简化系统设计----用文件大小除以块大小，能得出一个文件需要多少块

适合数据备份----以块，对数据进行冗余备份

### 名称节点--主节点

整个HDFS集群的管家，起到数据目录服务，负责整个文件系统元数据的存储，所有元数据信息都是保存在

内存中的

两个核心结构==》结合构造最新的元数据

- FslImage

用来保存系统文件树以及文件树中所有的文件和文件夹的元数据

维护文件的复制等级、修改和访问时间、访问权限、块大小以及组成文件的块

运行期间，静态保持不变

- EditLog

记录对数据进行的诸如创建、删除等操作

运气期间，不断增大

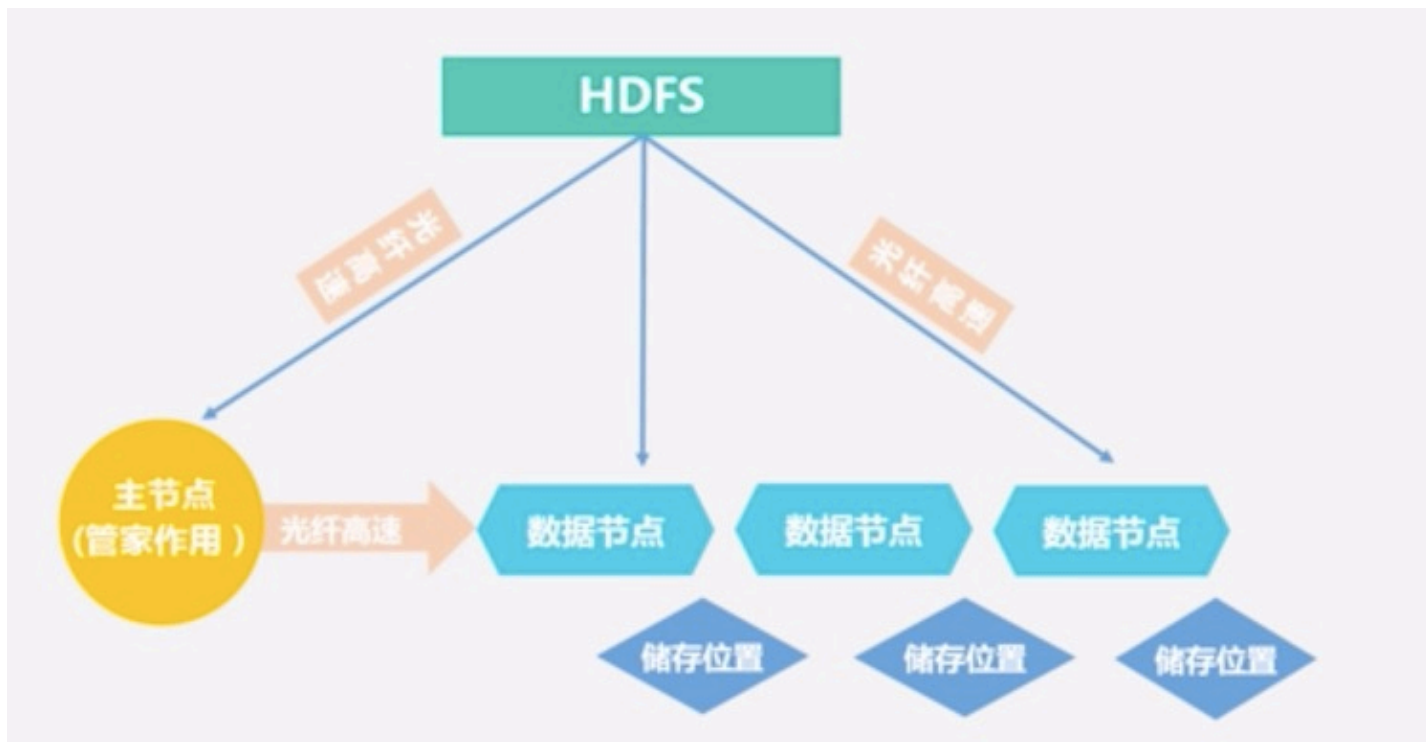
## 第二名称节点

- 解决EditLog不断增大，影响系统性能的问题
- 名称节点的冷备份

## 数据节点

负责具体数据的存储

## 4、体系结构



## HDFS命名空间

- 目录
- 文件
- 块

## HDFS1.0 局限性

- 命名空间限制：名称节点是保存在内存中。因此，名称节点能够容纳的对象的个数会受到空间大小的限制
- 性能的瓶颈：整个分布式文件的吞吐量，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，因此无法对不同应用程序进行隔离
- 集群的可用性：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

## 5、HDFS存储原理

### 冗余数据保存问题

HDFS中，每个数据都会被冗余保存，以块为单位。

### 冗余存储好处

- 加快数据传输速度
- 很容易检查数据错误

- 保证数据可靠性

## 数据保存策略问题

假设副本3份

### 数据存储

- 第一份副本：

放在上传文件的数据节点or随机挑选磁盘不太满，CPU不太忙的节点

- 第二份副本：

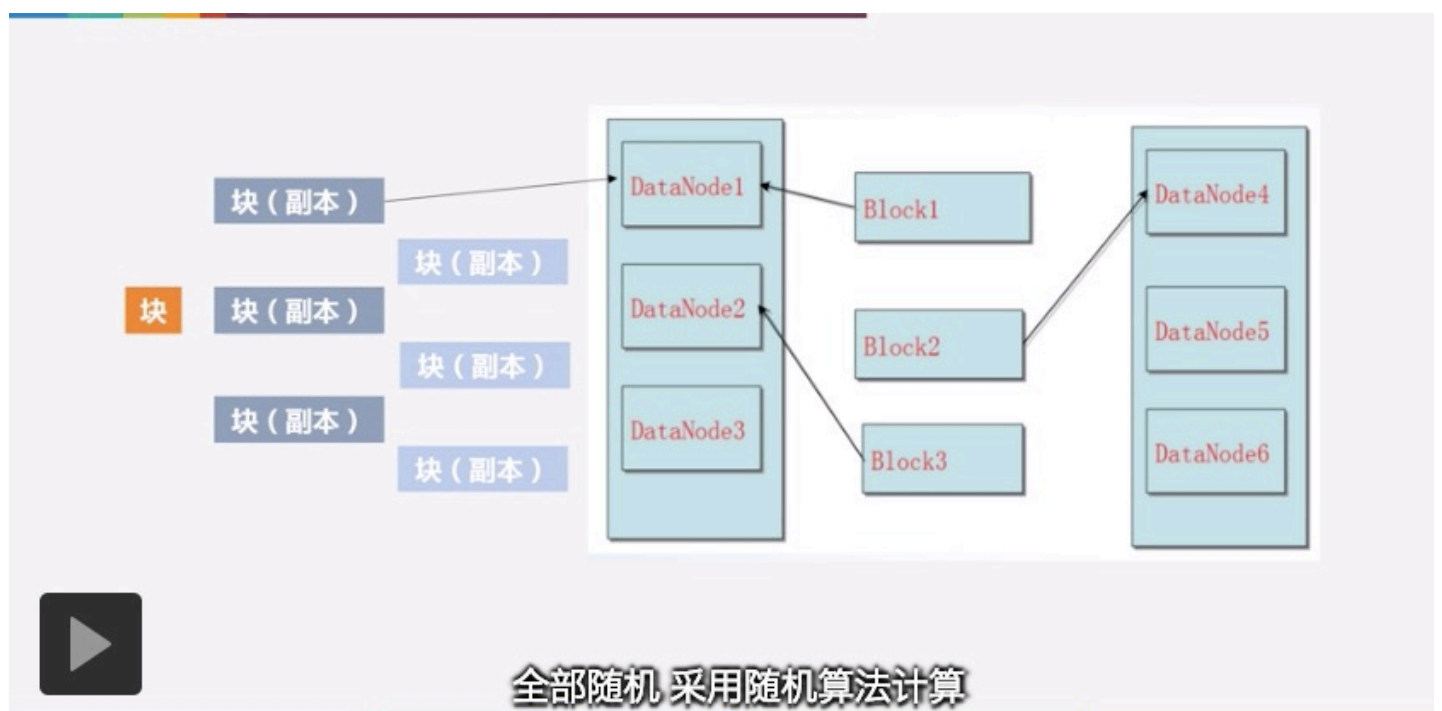
放在和第一个副本不同机架的节点上

- 第三份副本：

放在第一个副本相同机架的其他节点上

- 其余副本：

全部随机，采用随机算法计算



### 数据读取

- HDFS提供一个API可以确定一个数据节点所属的机架ID，客户端也可以调用API获取自己所属的机架ID
- 通过API算数据保存节点所属机架的ID，客户端也调用API去计算自身所在机架ID。若ID相同，则选这节

点，否则随机

## 数据恢复的问题

### 名称节点出错

- 冷备份
- 热备份

### 数据节点出错

如何知道数据节点发生错误？

数据节点在整个运行期间，都会定期的向名称节点发送心跳信息。一旦隔一个周期收不到心跳信息，则知数据节点发送故障


当名称节点知道该数据节点不可用，则会在其状态列表中标记数据节点不可用。把凡是存储在故障机上的数据，重新复制分发到其他正常可用的机器上。

===》HDFS优点：可以不断调整冗余数据的存储位置

### 数据本身出错

通过校验码，知道该数据是否出现问题。

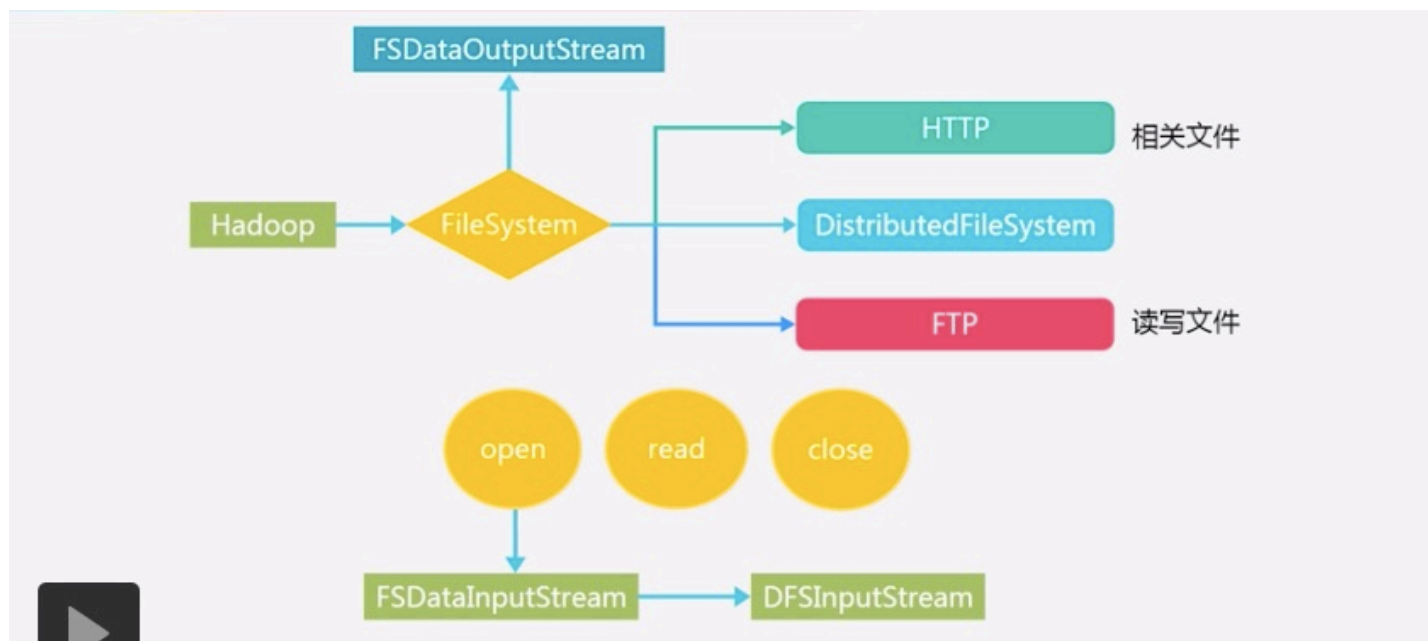
## 6、数据读写过程



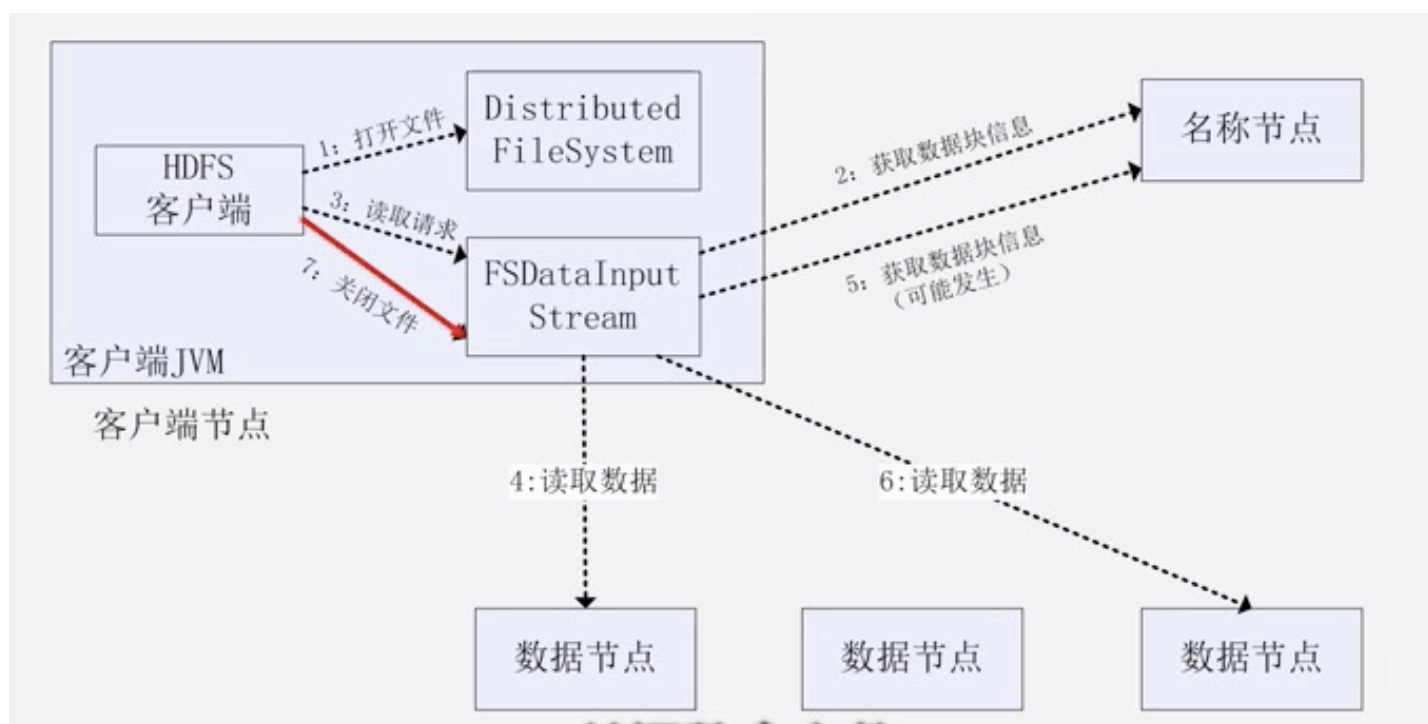
```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.fs.FSDDataInputStream;
public class Chapter3 {
    public static void main(String[] args) {
        try {
            Configuration conf = new Configuration();
            FileSystem fs = FileSystem.get(conf);
            Path filename = new Path("hdfs://localhost:9000/user/hadoop/test.txt");
            FSDDataInputStream is = fs.open(filename);
            BufferedReader d = new BufferedReader(new InputStreamReader(is));
            String content = d.readLine(); //读取文件一行
            System.out.println(content);
            d.close(); //关闭文件
            fs.close(); //关闭hdfs
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

现在要讲整个简单代码段背后

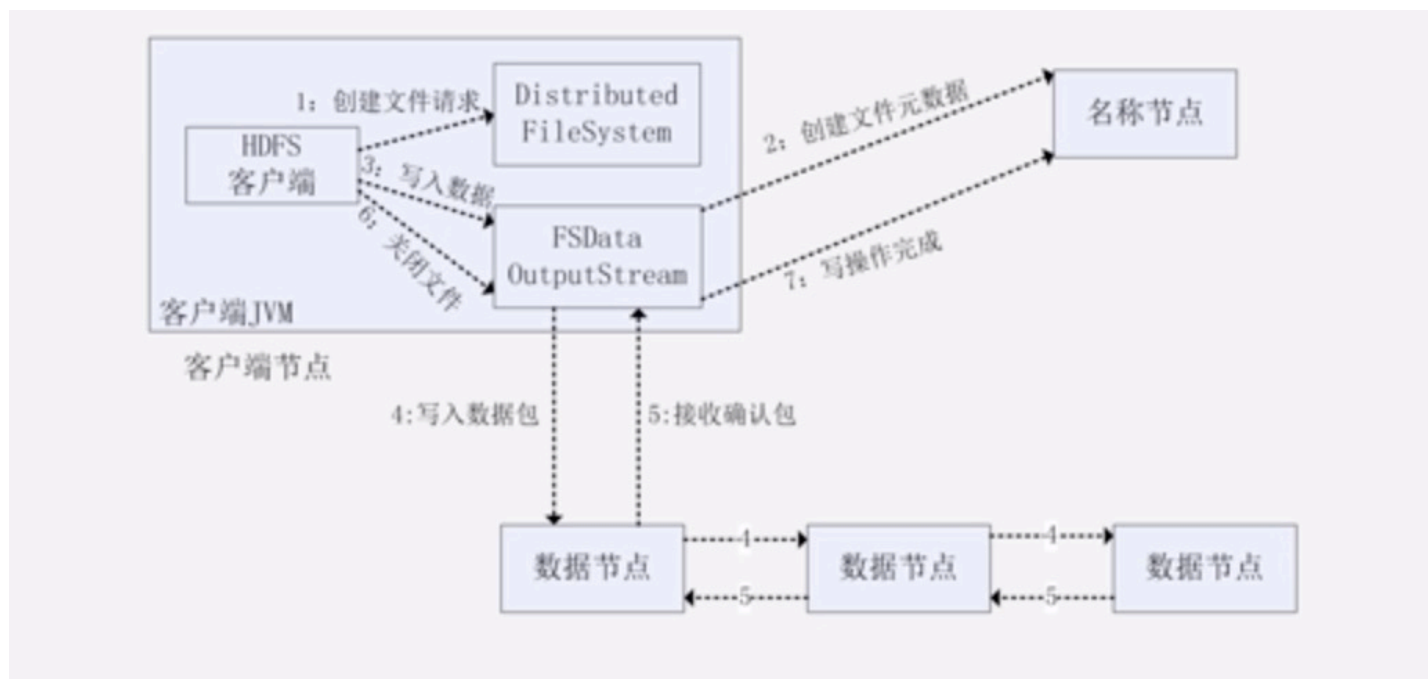
FileSystem：通用文件系统抽象基类



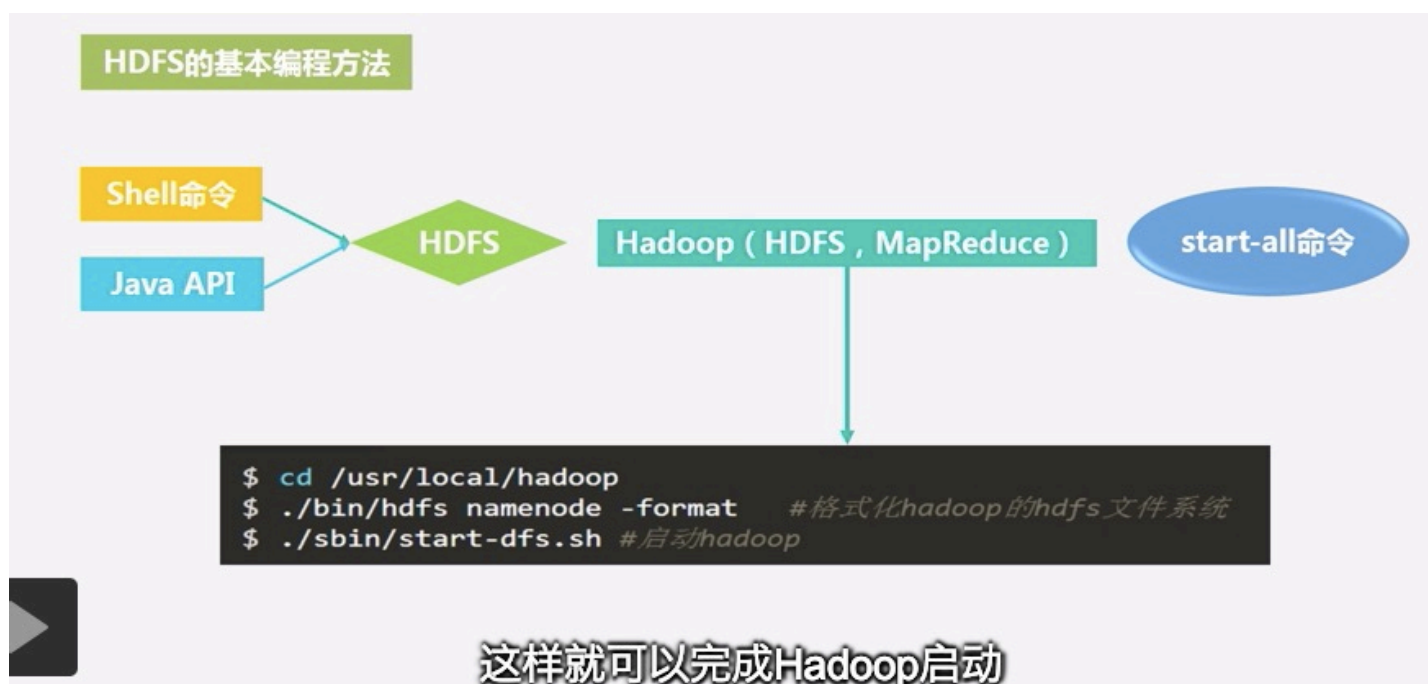
### 如何读数据

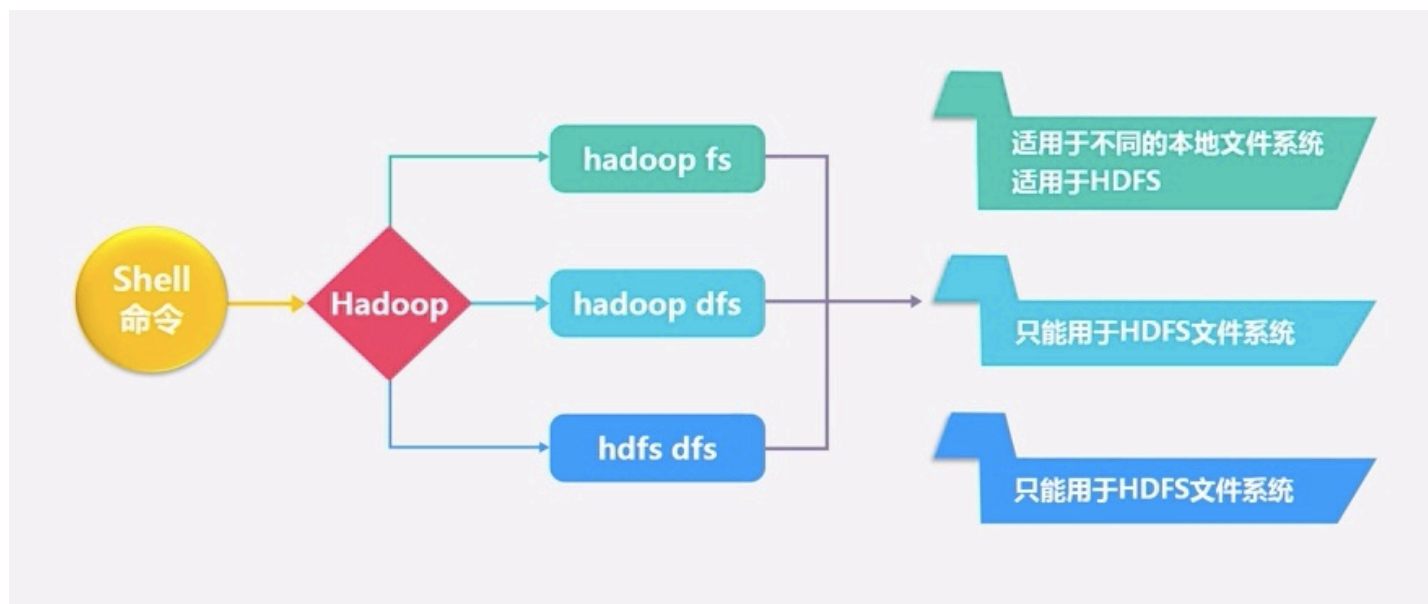


### 如何写数据

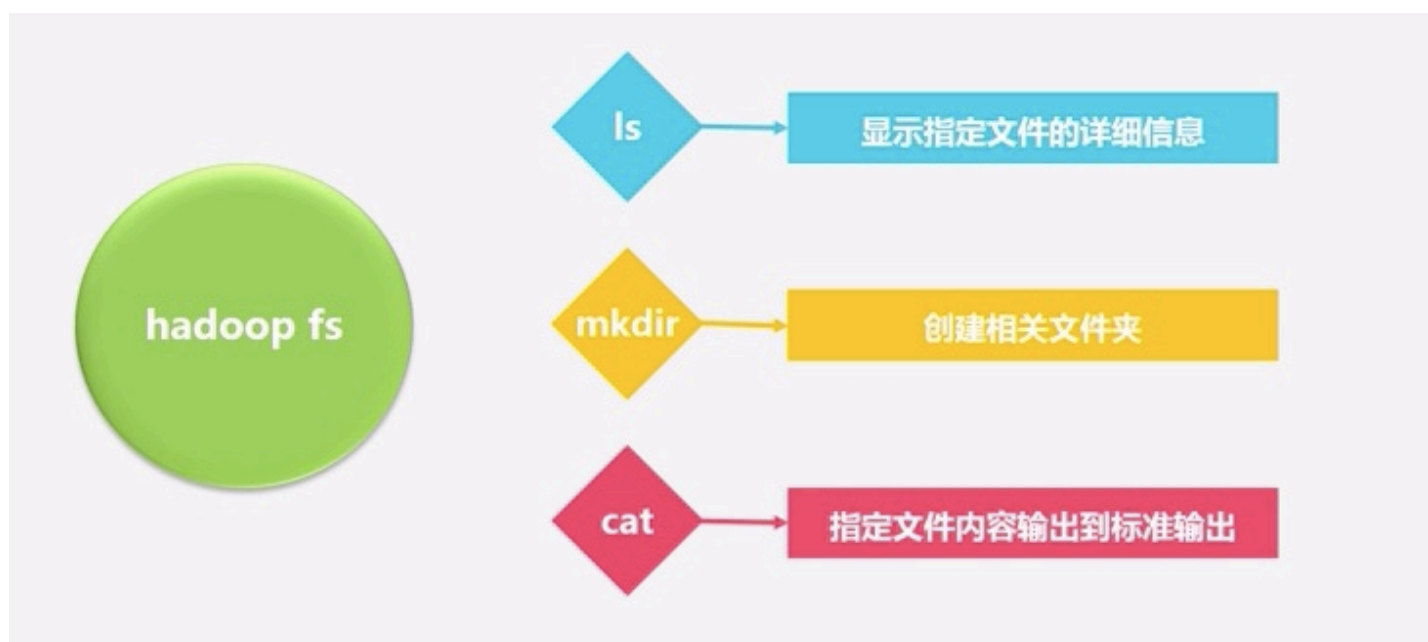


## 7、HDFS编程实践

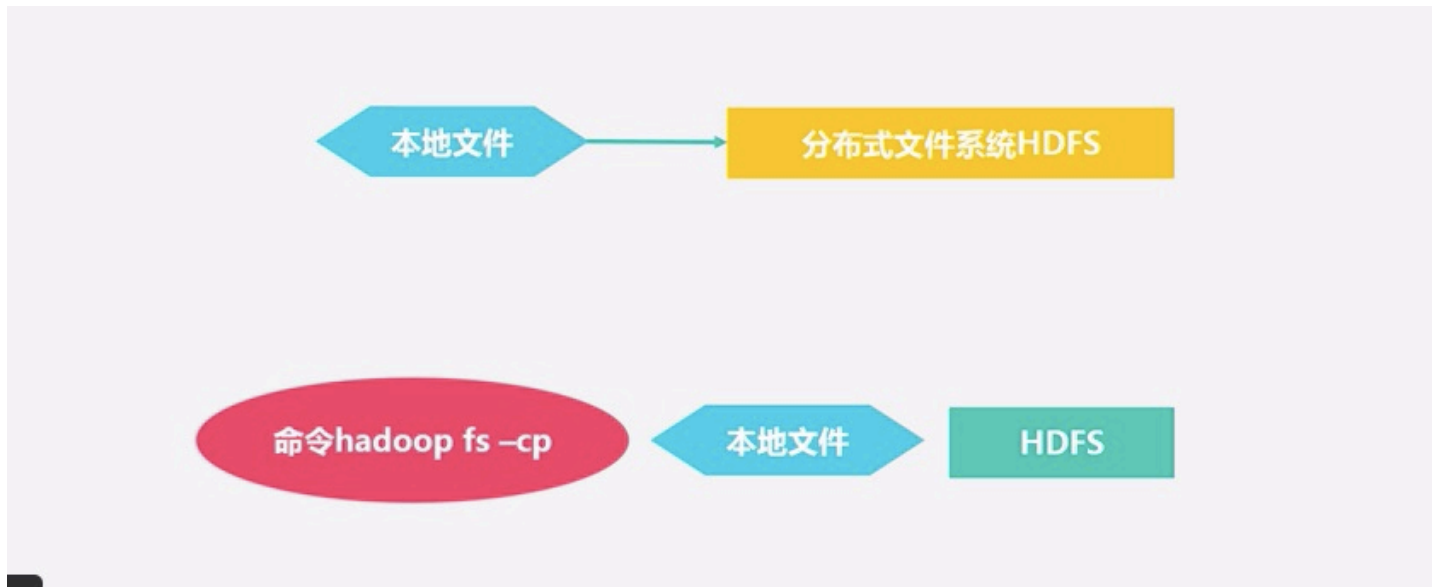




## HDFS常用命令







Java API与HDFS进行交互