HDFS 是 hadoop 的分布式文件系统，全称：Hadoop Distributed Filesystem。由 1 个 master(call me NameNode)和 N 个 slaver 组成（call me datanode）。其中 namenode 负责存储元数据,控制和协调 datanode 存储文件数据。通过写多份（可定义，默认 1）的方式实现数据的可靠性和读取的高效性。

主要特点：

1. 适合存储大文件，对海量小文件效率较低。这主要是由于存储在 namenode 上的大量小文件的文件元数据会让 namenode 成为瓶颈。
2. 标准流式访问。一次写入，多次读取是最高效的访问模式，只支持文件的追加写，不支持随机访问。
3. 对数据节点的硬件要求低，可靠性高，单台或多台节点故障一般不会中断服务（只要不是文件所在的所有副本存放节点都故障）
4. 适合做大数据量的离线分析，不适合做第时延的联机事务业务访问。

HDFS 的数据块

HDFS 的数据块（block）是该文件系统的最小读写单位，默认 64M（本地磁盘文件系统一般为 512K）之所以设置为比较大的块，目的是最小化寻址时间，使文件系统的传输速度尽可能的取决于磁盘本身的性能。

HDFS 的命令

Hdfs 和其它文件系统一样，提供了命令行方式操作和访问的功能。可以通过命令：hadoop fs 进行操作，而且比较简单和容易理解。

常用命令：

```
$ hadoop fs  -ls <path>
$ hadoop fs  -lsr <path> //递归
$ hadoop fs  -mkdir <path>
$ hadoop fs  -put <localsrc> ... <dst>
$ hadoop fs  -cp <src> <dst>
$ hadoop fs  -mv <src> <dst>
$ hadoop fs  -rm <path>
$ hadoop fs  -rmr <path> //递归
$ hadoop fs  -cat <src>

//查看文件系统的命令帮助
$ hadoop fs
//查看具体命令的使用帮助
$ hadoop fs  -help <cmd>
```

如：

```
[hadoop@hadoop00 ~]$ hadoop fs -help mv
11/10/24 09:59:50 INFO security.Groups: Group mapping
impl=org.apache.hadoop.security.ShellBasedUnixGroupsMapping; cacheTimeout=300000
-mv <src> <dst>:     Move files that match the specified file pattern <src>
                          to a destination <dst>.   When moving multiple files, the
                          destination must be a directory.
```

Hadoop 文件系统

Hadoop 有一个抽象的文件系统概念，抽象类：org.apache.hadoop.fs.FileSystem，HDFS 只是其中的一个实现。Mapreduce 理论上可以运行在任何一个实现中。实现类包括：

LocalFileSystem:本地文件系统

DistributedFileSystem:HDFS 分布式文件系统

HftpFileSystem:通过 HTTP 提供 HDFS 只读访问的文件系统

HsftpFileSystem: 通过 HTTPS 提供 HDFS 只读访问的文件系统

HarFileSystem:HDFS 归档文件系统

kosmosFileSystem:CloudStore 文件系统

FTPFileSystem:由 FTP 提供支持的文件系统

NativeS3FileSystem 和 S3FileSystem:由 Amazon S3 支持的文件系统

HDFS 的 JAVA-API

HDFS 分布式文件系统的 JAVA-API 提供了丰富的访问接口。主要包括：目录的创建，列表，查询，删除和文件的创建(写入)，读取等。这个就不好文字记录了，还是代码表达吧，热，快 1 点了，看来只有明天写了~~

Java 代码

```java
package org.acooly.hadoop.study;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.URI;
import java.net.URL;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FsUrlStreamHandlerFactory;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IOUtils;
import org.apache.log4j.Logger;

@SuppressWarnings("deprecation")
public class HDFSSample {

        static final Logger logger = Logger.getLogger(HDFSSample.class);
        static {
```

```
28.                 URL.setURLStreamHandlerFactory(new  FsUrlStreamHandlerFactory());
29.         }
30.
31.         public  static  void  main(String[]  args)  throws  Exception  {
32.                 HDFSSample  sample  =  new  HDFSSample();
33.                 String  cmd  =  args[0];
34.                 String  localPath  =  args[1];
35.                 String  hdfsPath  =  args[2];
36.                 if  (cmd.equals("create"))  {
37.                         sample.createFile(localPath,  hdfsPath);
38.                 }  else  if  (cmd.equals("get"))  {
39.                         boolean  print  =  Boolean.parseBoolean(args[3]);
40.                         sample.getFile(localPath,  hdfsPath,  print);
41.                 }
42.         }
43.
44.         /**
45.           *  创建文件
46.           *
47.           *  @param  localPath
48.           *  @param  hdfsPath
49.           *  @throws  IOException
50.           */
51.         @SuppressWarnings("deprecation")
52.         public  void  createFile(String  localPath,  String  hdfsPath)  throws  IOException  {
53.                 InputStream  in  =  null;
54.                 try  {
55.                         Configuration  conf  =  new  Configuration();
56.                         FileSystem  fileSystem  =  FileSystem.get(URI.create(hdfsPath),  conf);
57.                         FSDataOutputStream  out  =  fileSystem.create(new  Path(hdfsPath));
58.                         in  =  new  BufferedInputStream(new  FileInputStream(new  File(localPat
    h)));
59.                         IOUtils.copyBytes(in,  out,  4096,  false);
60.                         out.hsync();
61.                         out.close();
62.                         logger.info("create  file  in  hdfs:"  +  hdfsPath);
63.                 }  finally  {
64.                         IOUtils.closeStream(in);
65.                 }
66.         }
67.
68.         /**
69.           *  从 HDFS 获取文件
70.           *
```

```
71.          *  @param  localPath
72.          *  @param  hdfsPath
73.          *  @throws  IOException
74.          */
75.         public  void  getFile(String  localPath,  String  hdfsPath,  boolean  print)  throws  IOExc
    eption  {
76.
77.                 Configuration  conf  =  new  Configuration();
78.                 FileSystem  fileSystem  =  FileSystem.get(URI.create(hdfsPath),  conf);
79.                 FSDataInputStream  in  =  null;
80.                 OutputStream  out  =  null;
81.                 try  {
82.                         in  =  fileSystem.open(new  Path(hdfsPath));
83.                         out  =  new  BufferedOutputStream(new  FileOutputStream(new  File(localPat
    h)));
84.                         IOUtils.copyBytes(in,  out,  4096,  !print);
85.                         logger.info("get  file  form  hdfs  to  local:  "  +  hdfsPat
    h  +  ",  "  +  localPath);
86.                         if  (print)  {
87.                                 in.seek(0);
88.                                 IOUtils.copyBytes(in,  System.out,  4096,  true);
89.                         }
90.                 }  finally  {
91.                         IOUtils.closeStream(out);
92.                 }
93.         }
94. }
```

运行测试代码

可以直接在 WINDOWS 本机运行，通过 HADOOP 客户端 API 访问 HDFS:
//创建文件

java org.acooly.hadoop.study.HDFSSample create D:/temp/keepalive.txt
hdfs://hadoop00:9000/input/keepalive.txt

//读取文件

java org.acooly.hadoop.study.HDFSSample get hdfs://hadoop00:9000/input/keepalive.txt
D:/temp/keepalive_get.txt