本课暗号：明月别枝惊鹊，清风半夜鸣蝉

简答：hdfs上的每个块，为什么要存储多个副本？

答:

这就是HDFS受欢迎的原因,因为HDFS集群不需要搭建在高标准或者其他高要求的服务器上面,可以使用普通的商业服务器,但是普通是商业服务器存在稳定性不够好的问题,那么保存多个block副本,有助于提高容错率,从而降低因服务器宕机而导致数据丢失的情况出现.

HDFS将文件以block的形式进行拆分,每个block分别保存在不同的DataNode服务器结点上面.

自主完成hdfs文件删除操作

|  |
| --- |
| @Test //hdfs文件删除操作 public void deleteFileForHdfs() throws IOException {  //获取文件系统  Configuration configuration = new Configuration();  configuration.set("fs.defaultFS", "hdfs://node01:8020");  FileSystem fileSystem = FileSystem.*get*(configuration);  //执行文件删除  fileSystem.deleteOnExit(new Path("/nangua/1.txt"));  //关闭资源  fileSystem.close(); } |

自主完成hdfs文件重命名操作

|  |
| --- |
| @Test //hdfs文件重命名操作 public void mvFileForHdfs() throws IOException {  //获取文件系统  Configuration configuration = new Configuration();  configuration.set("fs.defaultFS", "hdfs://node01:8020");  FileSystem fileSystem = FileSystem.*get*(configuration);  //执行重命名  fileSystem.rename(new Path("/nangua/test1"), new Path("/nangua/test"));  //关闭资源  fileSystem.close(); } |

查看hdfs文件相信信息

|  |
| --- |
| @Test //查看hdfs文件相信信息  public void appendToHdfs() throws IOException, URISyntaxException {  /\*  \* 查看hdfs文件相信信息  \* 1获取文件系统  \* 2 获取文件详情  \* // 输出详情 \文件名称 \长度 \分组 \获取存储的块信息 \获取块存储的主机节点  \* 3 关闭资源  \*/  Configuration configuration = new Configuration();  FileSystem fileSystem = FileSystem.*get*(new URI("hdfs://node01:8020"),configuration);   RemoteIterator<LocatedFileStatus> fileList = fileSystem.listFiles(new Path("/nangua"),false);  while (fileList.hasNext()){  LocatedFileStatus status = fileList.next();  System.*out*.println(status.getPath());//查看文件路径  System.*out*.println(status.getPath().getName());//查看文件名 // System.out.println(status.getPath().getParent());//获取文件所在文件夹 // System.out.println(status.getGroup());//获取组 // System.out.println(status.getLen());//获取文件长度 // System.out.println(status.getOwner());//获取文件拥有着 // System.out.println(status.getPermission());//获取权限信息 // System.out.println(status.getBlockSize());//获取块长度  BlockLocation[] blockLocations = status.getBlockLocations();//获取块信息  for(BlockLocation blockLocation : blockLocations){  String[] blocks = blockLocation.getHosts();//获取块的主机结点  for(String block:blocks){  System.*out*.println(block);  }  String[] names = blockLocation.getNames();//获取块的主机名称  for (String name:names){  System.*out*.println(name);  }  }  }  fileSystem.close();//关闭资源   } |

自主实现通过IO流从hdfs上面下载文件

|  |
| --- |
| @Test //通过IO流完成数据下载 public void getFileFromHdfs() throws URISyntaxException, IOException {  //获取文件系统  Configuration configuration = new Configuration();  FileSystem fileSystem = FileSystem.*get*(new URI("hdfs://node01:8020"),configuration);  //创建本地输出流  FileOutputStream fos = new FileOutputStream("file1.txt");  //获取输入流  FSDataInputStream fis = fileSystem.open(new Path("/nangua/file1.txt"));  //流对拷  IOUtils.*copy*(fis,fos);  //关闭资源  IOUtils.*closeQuietly*(fis);  IOUtils.*closeQuietly*(fos);  fileSystem.close(); } |

hdfs的小文件合并上传

|  |
| --- |
| @Test //小文件合并上传处理 public void mergeFileToHdfs() throws URISyntaxException, IOException {  //获取文件系统  Configuration configuration = new Configuration();  FileSystem fileSystem = FileSystem.*get*(new URI("hdfs://node01:8020"),configuration);  //获取输出流  FSDataOutputStream fos = fileSystem.create(new Path("/nangua/newFile.txt"));   //获取本地文件系统 localFileSystem  LocalFileSystem localFileSystem = FileSystem.*getLocal*(configuration);  //读取本地的文件  FileStatus[] fileStatuses = localFileSystem.listStatus(new Path("file:///C:\\Users\\Thinkpad\\Documents\\smallfile"));  for(FileStatus fileStatus : fileStatuses){  //获取文件路径  Path path = fileStatus.getPath();  //读取文件,获取输入流  FSDataInputStream fis = localFileSystem.open(path);  IOUtils.*copy*(fis,fos);  IOUtils.*closeQuietly*(fis);  }  //关闭资源  IOUtils.*closeQuietly*(fos);  fileSystem.close();  localFileSystem.close(); } |

将hdfs的文件合并后下载

|  |
| --- |
| @Test //合并hdfs系统文件下载 public void mergeFileToLocal() throws URISyntaxException, IOException {  //获取文件系统  Configuration configuration = new Configuration();  FileSystem fileSystem = FileSystem.*get*(new URI("hdfs://node01:8020"),configuration);  //获取输出流  FileOutputStream fos = new FileOutputStream("result.txt");  //遍历目标路径  FileStatus[] fileStatuses = fileSystem.listStatus(new Path("/nangua"));  for(FileStatus fileStatus : fileStatuses){  //读取文件  Path path = fileStatus.getPath();  //获取输入流  FSDataInputStream fis = fileSystem.open(path);  //读取文件  IOUtils.*copy*(fis,fos);  IOUtils.*closeQuietly*(fis);  }  IOUtils.*closeQuietly*(fos);  fileSystem.close(); } |