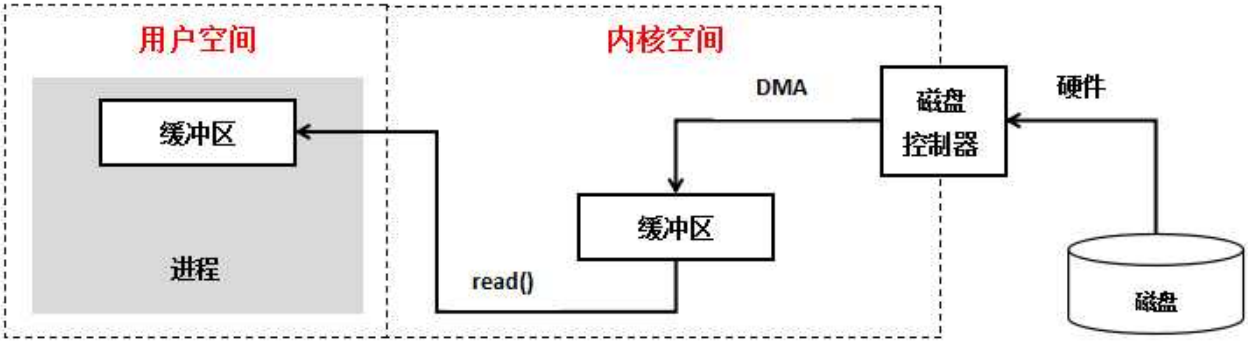
高频面试题：什么是零拷贝？在那些地方使用了。

第一部分：零拷贝原理剖析

一、几个重要的概念

1.1 用户空间和内核空间

操作系统的核心是内核，独立于普通的应用程序，可以访问受保护的内存空间，也有访问底层硬件设备的所有权限。为了保证用户进程不能直接操作内核 (kernel)，保证内核的安全，操作系统将虚拟空间划分为两部分，一部分为内核空间，一部分为用户空间。

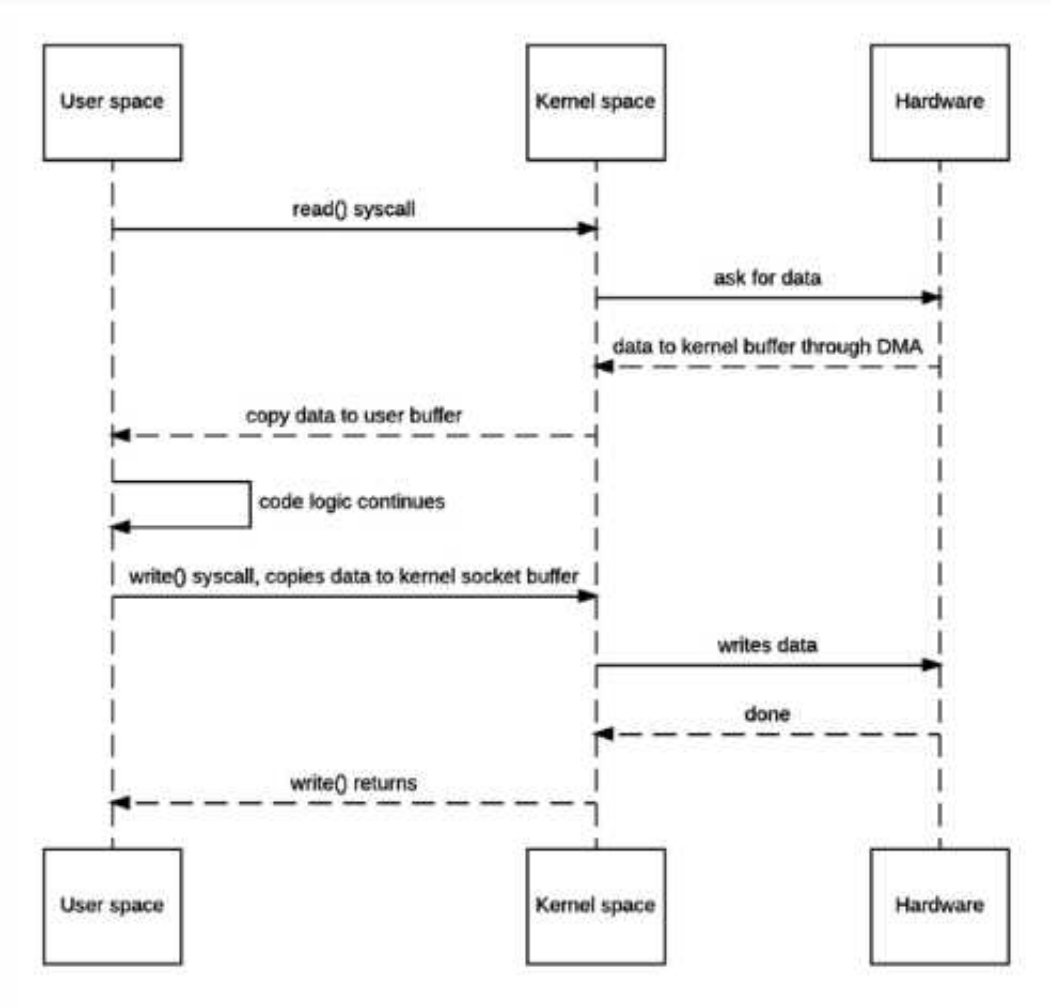


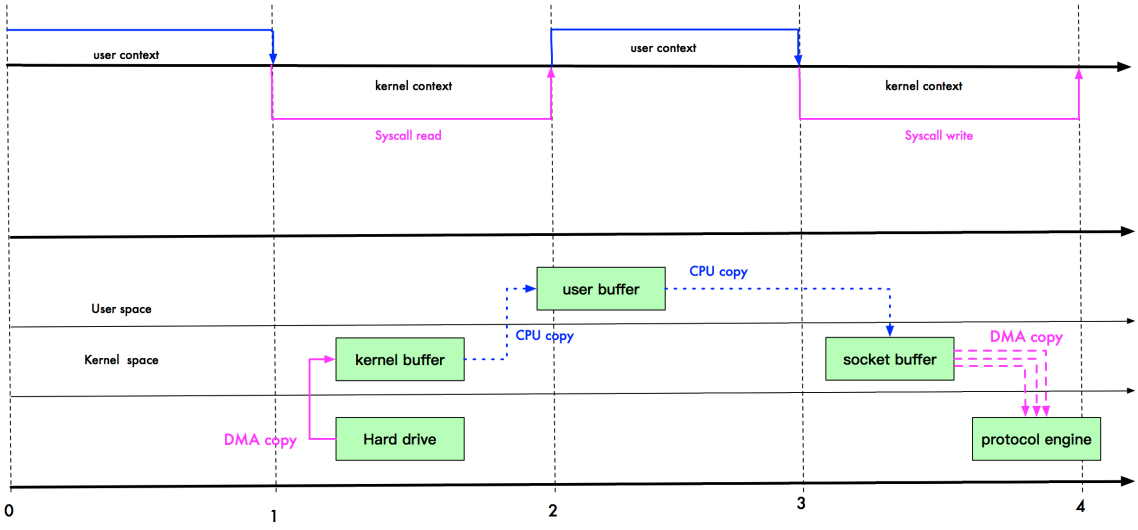
1.2 IO两个流程

第一阶段：等待网络上的数据分组到达，然后被复制到内核的某个缓冲区。

第二阶段：把数据从内核缓冲区复制到应用程序缓冲区中。

二、传统的风IO流





1. 通过DMA copy 数据从hard driver 拷贝到kernel buffer
2. 通过 CPU copy 数据从kernel buffer 拷贝到 user buffer
3. 通过 CPU copy 数据从 user buffer 拷贝到kernel buffer
4. 通过 CPU copy 数据从 kernel buffer 拷贝到socket buffer
5. 通过DMA copy 将socket buffer 中的数据发送出去

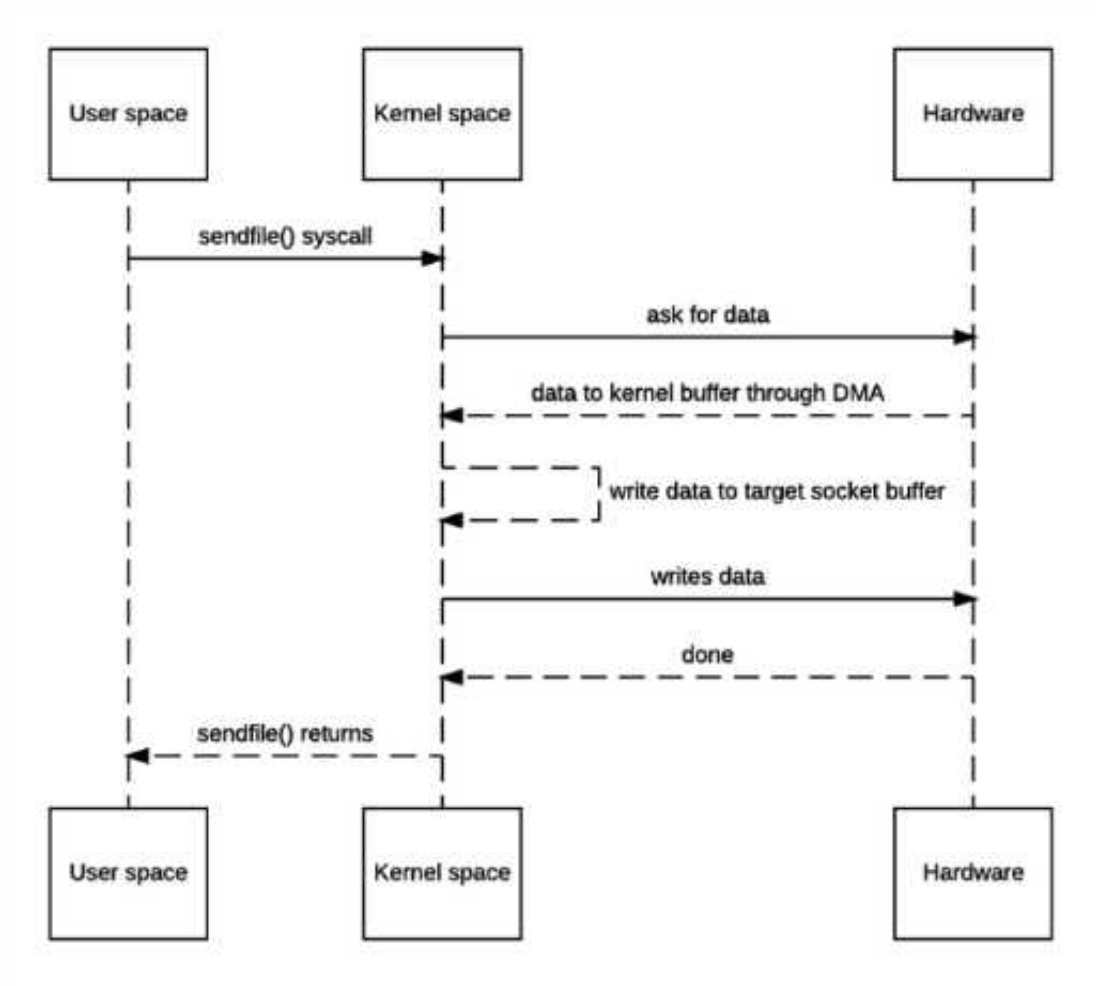
三、零拷贝（zero-copy）

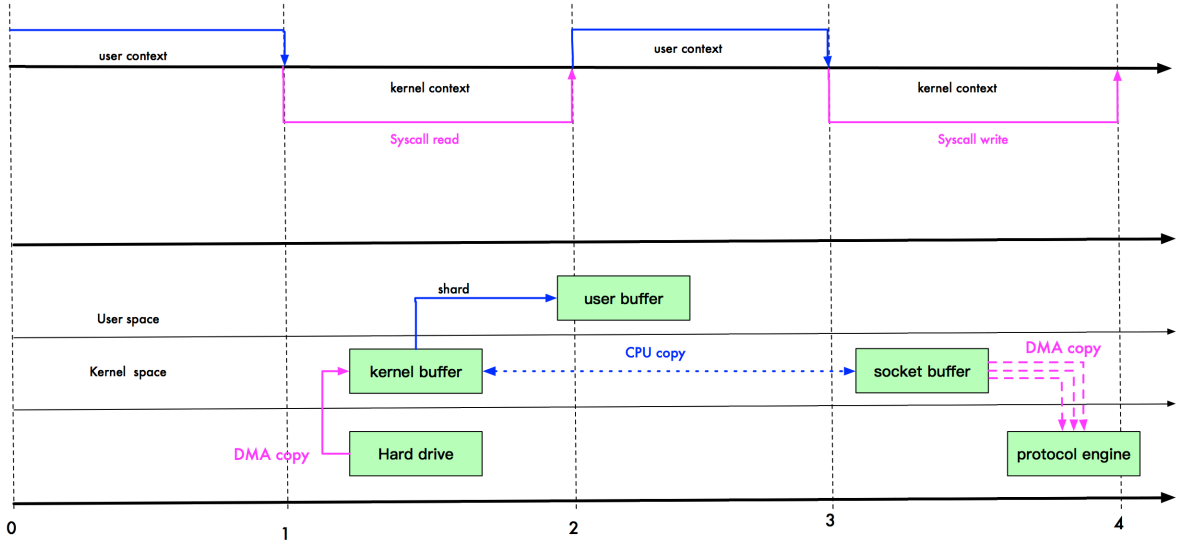
3.1 零拷贝的概念

零拷贝主要的任务是避免CPU将数据从一块存储拷贝到另外一块存储，主要就是利用各种拷贝的技术，避免让CPU做大量的数据拷贝任务，减少不必要的拷贝，或者说让别的组建来做这一类简单的数据传输，让CPU专注于别的任务。这样就可以让系统资源利用的更加有效。

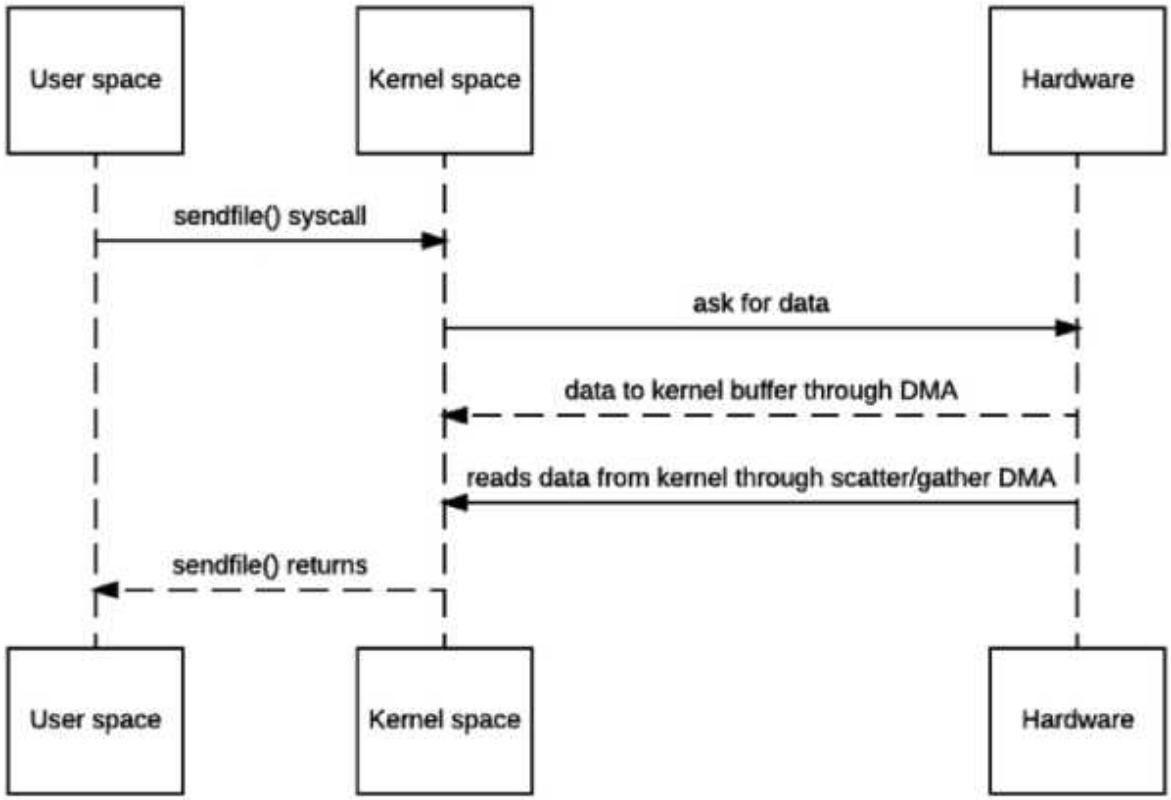
如何减少数据拷贝的次数呢？减少用户空间到内核空间的拷贝 mmap 优化。

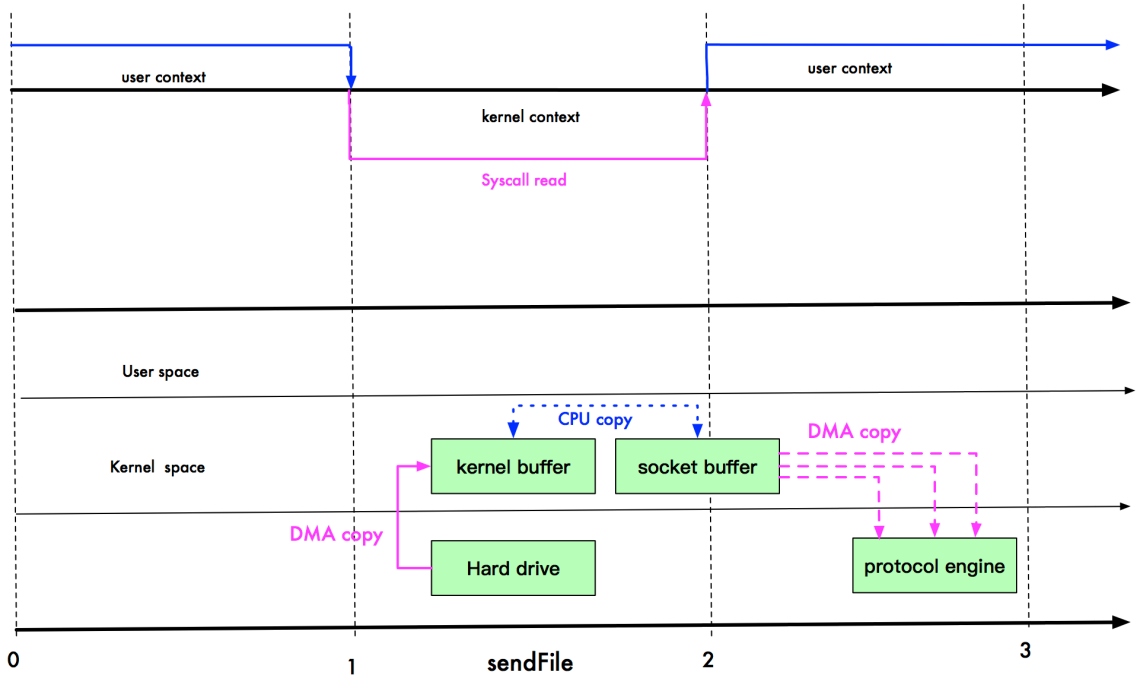
3.2 减少用户空间和内核空间的拷贝





3.3 直接传递文件描述（sendfile）





四、零拷贝的再次理解

1）零拷贝是从操作系统角度来说的，应为内核缓冲区之前，没有数据的重复（只有kernel buffer 有一份数据）。

2）零拷贝不仅仅带来了更少的数据复制，还能带来其他的性能优势，例如：更少的上下文切换，更少的CPU缓存伪共享以及无CPU校验和计算。

五、mmap 和sendFile 的区别

1. mmap 适合小数据量读写，sendFile适合大文件传输
2. mmap 需要4次上下文切换、3次数据拷贝；sendFile 需要3次上下文切换、最少2次数据拷贝。
3. sendFile 可以利用DMA方式，减少CPU拷贝，mmap 则不能（必须从内核拷贝到socket 缓冲区）。

在这个选择上：RocketMQ 在消费消息时，使用了mmap。kafka使用了sendFIle。

六、哪些地方会使用到零拷贝技术

场景：

文件较大，读写较慢，追求速度

JVM内存不够，不能加载太大的数据

内存宽带不够，即存在其他程序或线程存在大量的IO操作，导致带宽本来就小

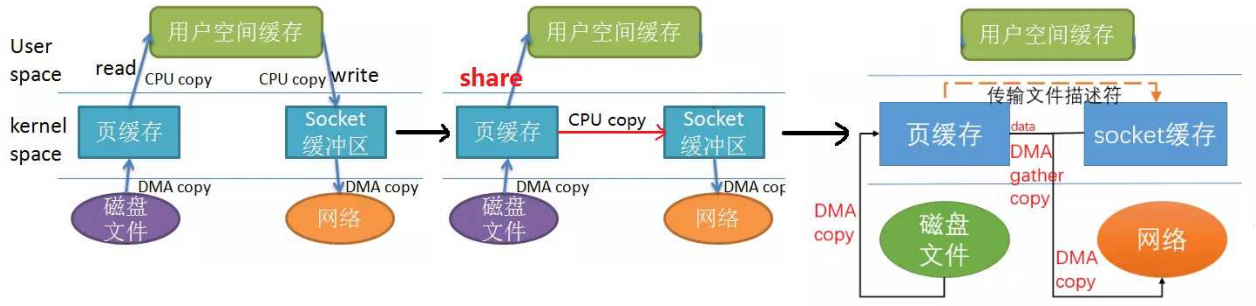
技术：

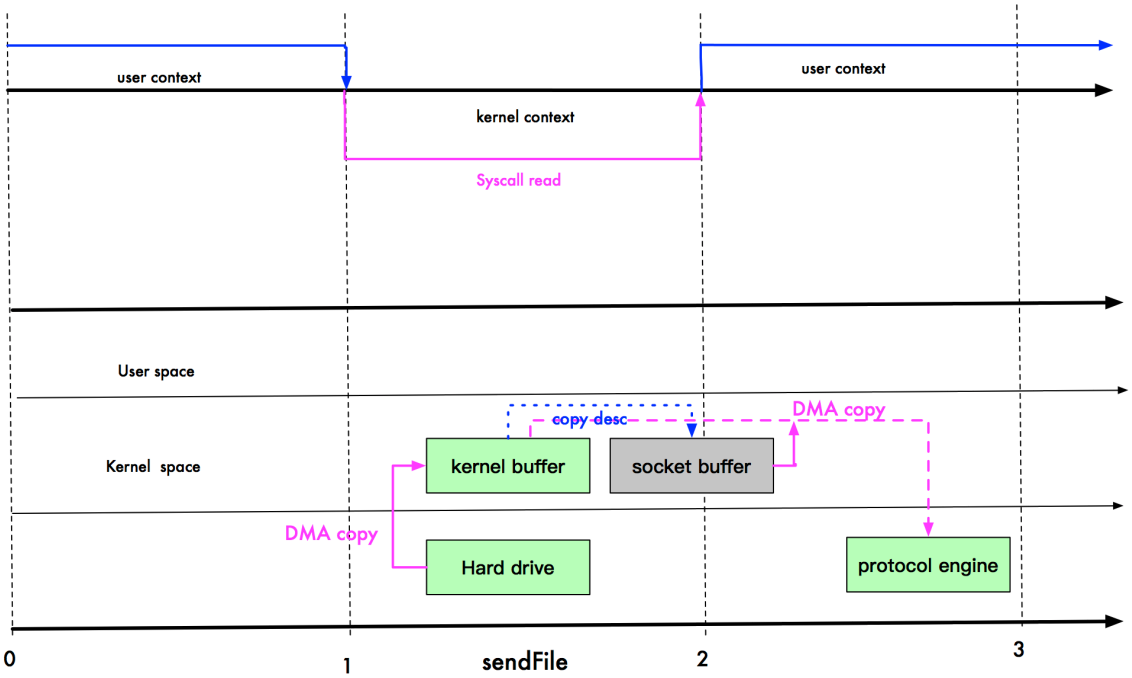
Java的NIO

Netty

kafka

七、总结





第二部分：零拷贝案例深度剖析

一、编写传统IO文件读写案例

1.1 服务端代码

```
package org.wdzl.zerocopy;

import java.io.DataInputStream;
```

```
import java.net.ServerSocket;
import java.net.Socket;

/**
 * 传统IO读写案例的服务端
 */
public class TranditionServer {
    public static void main(String[] args) throws  Exception {
        //创建serversocket 对象
        ServerSocket serverSocket = new ServerSocket(8089);
        //循环监听连接
        while (true){
            Socket socket = serverSocket.accept();
            //创建输入流对象
            DataInputStream dataInputStream = new
DataInputStream(socket.getInputStream());
            try{
                //创建缓冲区字节数组
                byte[] bytes = new byte[1024];
                while(true){
                    int readCount = dataInputStream.read(bytes, 0,
bytes.length);
                    if(readCount==-1){
                        break;
                    }
                }
            }catch (Exception e){
                e.printStackTrace();
            }

        }


    }
}
```

accept() 源码解读

```
 /**
       监听者向socket发送链接并且接受这个链接
     * Listens for a connection to be made to this socket and accepts
       此方法会阻塞直接到链接连接建立起，执行最后返回
     * it. The method blocks until a connection is made.
     *
     * <p>A new Socket {@code s} is created and, if there
     * is a security manager,
     * the security manager's {@code checkAccept} method is called
     * with {@code s.getInetAddress().getHostAddress()} and
     * {@code s.getPort()}
```

```
     * as its arguments to ensure the operation is allowed.
     * This could result in a SecurityException.
     *
     * @implNote
     * An instance of this class using a system-default {@code SocketImpl}
     * accepts sockets with a {@code SocketImpl} of the same type, regardless
     * of the {@linkplain Socket#setSocketImplFactory(SocketImplFactory)
     * client socket implementation factory}, if one has been set.
     *
     * @throws     IOException  if an I/O error occurs when waiting for a
     *               connection.
     * @throws     SecurityException  if a security manager exists and its
     *             {@code checkAccept} method doesn't allow the operation.
     * @throws     SocketTimeoutException if a timeout was previously set with
setSoTimeout and
     *             the timeout has been reached.
     * @throws     java.nio.channels.IllegalBlockingModeException
     *             if this socket has an associated channel, the channel is in
     *             non-blocking mode, and there is no connection ready to be
     *             accepted
     *
     * @return the new Socket
     * @see SecurityManager#checkAccept
     * @revised 1.4
     * @spec JSR-51
     */
    public Socket accept() throws IOException {
        if (isClosed())
            throw new SocketException("Socket is closed");
        if (!isBound())
            throw new SocketException("Socket is not bound yet");
        Socket s = new Socket((SocketImpl) null);
        implAccept(s);
        return s;
    }
```

1.2 客户端代码

```
package org.wdzl.zerocopy;

import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.InputStream;
import java.net.Socket;

/**
 * 传统io读写客户端
```

```
 */
public class TranditionClient {
    public static void main(String[] args) throws  Exception {
        //创建socket对象
        Socket socket = new Socket("localhost",8089);
        //创建磁盘文件
        String fileName = "/Users/lily/Desktop/5.png";
        //创建输入流对象
        InputStream inputStream = new FileInputStream(fileName);
        //创建输出流
        DataOutputStream dataOutputStream = new
DataOutputStream(socket.getOutputStream());
        //创建字节数组
        byte[] buffer = new byte[1024];
        long readCount = 0;
        long total=0;
        long startTime = System.currentTimeMillis();
        while ((readCount=inputStream.read(buffer))>=0){
            total+=readCount;
            dataOutputStream.write(buffer);
        }
        long endTime = System.currentTimeMillis();
        System.out.println("发送总字节数: "+total+", 耗时: "+(endTime-
startTime)+"ms");
        //释放资源
        dataOutputStream.close();
        socket.close();
        inputStream.close();
    }
}
```

采用零拷贝后的IO文件读写

1.3 新的IO服务端源码

```
package org.wdzl.zerocopy;

import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.nio.ByteBuffer;
import java.nio.channels.ServerSocketChannel;
import java.nio.channels.SocketChannel;

public class NewIOServer {
    public static void main(String[] args) throws Exception {
        InetSocketAddress address = new InetSocketAddress(8089);
        ServerSocketChannel serverSocketChannel = ServerSocketChannel.open();
        ServerSocket serverSocket = serverSocketChannel.socket();
        serverSocket.setReuseAddress(true);
```

```
        serverSocket.bind(address);


        ByteBuffer byteBuffer = ByteBuffer.allocate(1024);
        while (true){
            SocketChannel socketChannel = serverSocketChannel.accept();
            socketChannel.configureBlocking(true);
            int readCount = 0;
            while (readCount!=-1){
                readCount = socketChannel.read(byteBuffer);
                byteBuffer.rewind();
            }

        }
    }
}
```

1.4 新的io客户端源码

```
package org.wdzl.zerocopy;

import java.io.FileInputStream;
import java.net.InetSocketAddress;
import java.nio.channels.FileChannel;
import java.nio.channels.SocketChannel;

public class NewIOClient {
    public static void main(String[] args) throws  Exception {
        SocketChannel socketChannel = SocketChannel.open();
        socketChannel.connect(new InetSocketAddress("localhost",8089));
        socketChannel.configureBlocking(true);
        String fileName = "/Users/lily/Desktop/5.png";
        FileChannel fileChannel = new FileInputStream(fileName).getChannel();
        long startTime = System.currentTimeMillis();

        //transferTo 方法用到了零拷贝，操作系统为其提供的特性
        long transferCount = fileChannel.transferTo(0, fileChannel.size(),
socketChannel);
        long endTime   = System.currentTimeMillis();
        System.out.println("发送总字节数：+transferCount+"耗时：+(endTime-
startTime)+"ms");
        //释放资源
        fileChannel.close();
    }
}
```