Java 非阻塞 I/O 在乡村可视化远程医疗系统中的应用*

顾和明
(长江师范学院 数学与计算机学院,重庆 408100)

摘 要:分析了乡村可视化远程医疗系统采用多线程技术实现网络通信存在的不足,阐述了 Java 非阻塞 I/O 的基本原理。系统采用非阻塞 I/O 通信技术只使用一个线程并行实现大量客户无阻塞的通信,有效地减少了系统开销,较好地提升了系统的性能。

关键词:非阻塞 I/O;远程医疗系统;线程阻塞;通道
中图分类号:TP393.09 **文献标识码**:B **文章编号**:1674-7720(2011)06-0092-03

Application of Java non-blocking I/O in remote country visualization medicine system

Gu Heming
(Mathematics and Computer Science Institute, Yangtze Normal University, Chongqing 408100, China)

Abstract: In this paper, disadvantages of using multithreading in remote country visualization medicine system and the fundamental principle of Java non-blocking I/O are expounded. With Java non-blocking I/O, we can develop the program working on the non-blocking model in one threading, which ensures the server to consume less resources and run more efficiently, thus provide parallel communication service for thousands of clients.

Key words: non-blocking; remote medicine system; thread blocking; channel

1 乡村可视化远程医疗系统的应用前景

目前我国广大的农村地区医疗条件比较差,这些地区的大量患者因为得不到及时有效的治疗,给患者以及家属带来了不必要的痛苦。使用网络多媒体通信技术实现的乡村可视化远程医疗系统可以让高水平的医疗专家给乡村的患者进行远程诊病,以及指导治疗与护理。乡村卫生所、乡镇医院等医疗单位可以和具有优质医疗资源的医院或直接与专家建立医疗合作关系,双方合作使用可视化医疗系统,让专家为病人进行远程诊疗服务。在乡村这一端,一般由专业医务人员指导病人接受专家远程诊疗,医疗专家通过可视化医疗系统实时对病人进行诊断、处方、治疗甚至指导手术等。患者也可以在家中使用远程医疗系统直接接受远程专家的指导,尤其在处理突发性疾病时,可以在第一时间内得到医疗专家的帮助。目前随着生活水平的大幅度提高,不少需要"私人医生"的人员也可以方便地使用该系统得到医疗专家的服务。可见,可视化医疗系统有着很广阔的应用市场和较好的发展前景。

2 乡村可视化远程医疗系统功能简介

乡村远程医疗系统可以同时提供若干对医疗服务,每一对服务有 2 个或 2 个以上的客户端进行通信。客户端分别由医疗专家和患者使用,医疗专家使用的客户端称为专家端,患者接受诊疗的客户端称为医务端。每个客户(医务端和专家端)首先登录服务器,服务器将每个登录客户的用户名、IP 地址等相关信息发送给相关的其他客户,然后客户端之间建立直接连接,并可进行文件传送、文字和多媒体等通信。通过这种综合的通信方式,医疗专家可以远程为患者提供医疗服务。为了建立医疗档案,服务器需要保存诊疗时间、参与人员、患者检验报告、病情诊断、处方等信息。患者还可以通过服务器进行

* 基金项目:国家教育部"春晖"计划(Z2005-1-55003)

专家预约，专家和患者均可以在服务器上进行诊疗信息查询。由于视频信息量比较大，且保存意义不是十分大，因此并不保存在服务器上。系统结构如图 1 所示。

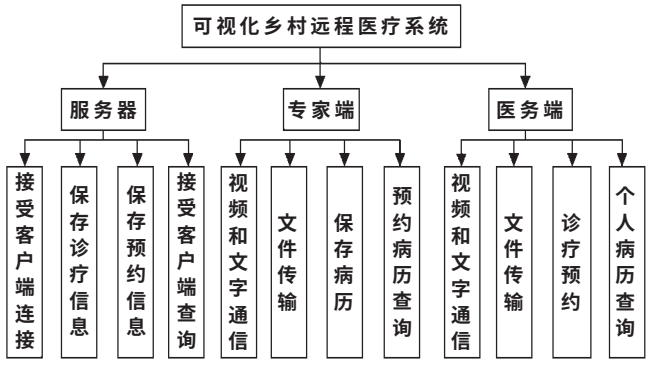

图 1 可视化乡村远程医疗系统功能结构

3 使用多线程和阻塞通信模式存在的局限

可视化远程医疗服务器需要同时为多个专家端和医务端提供服务，在传统的阻塞通信模式中使用多线程技术来处理多客户连接，一般需要服务器为每个客户端建立一个线程。但使用多线程技术存在以下局限：

（1）Java 虚拟机会为每个线程分配独立的堆栈空间，工作线程数目越多，系统开销就越大，而且增加了 Java 虚拟机调度线程的负担，增加了线程之间同步的复杂性，提高了线程死锁的可能性；

（2）当主线程接收客户连接，在医疗服务过程中，服务器从网络发送或接收数据时，线程常常进入阻塞状态，这就使工作线程的许多时间都浪费在阻塞操作上，CPU 的利用率降低，此外 Java 虚拟机需要频繁地转让 CPU 的使用权。

由此可见，工作线程并不是越多越好。实验表明，适量的工作线程会提高服务器的并发性能，但是当工作线程的数目达到某个极限而超出系统的负荷时，反而会降低并发性能，使得许多客户端得不到服务器的及时响应。为了改善服务器性能，在远程医疗系统中采用了 Java 非阻塞 I/O 通信技术。

4 Java 非阻塞 I/O 工作原理简介

在传统的阻塞通信模式中，服务器在接受客户连接后创建 Socket 对象与客户进行通信，Socket 对象由线程进行管理。当有多个客户连接时，就需要多个线程分别处理服务器与各个客户的通信。Java 非阻塞 I/O 通信服务器程序只需要一个线程就能够实现多个客户端的连接、同时接收多个客户端发送的数据，以及同时向多个客户端发送响应数据。这种非阻塞通信采用了基于 Channel（通道）、Selector（选择器）、Buffer（缓冲器）的新模式。

非阻塞 I/O 通信中 Channel 是一个接口，功能类似于传统 I/O 中的 Stream，但通道具有双向性，既可读入，也可写出。SocketChannel 和 ServerSocketChannel 均可实现 Channel 接口，它们是 Socket 和 ServerSocket 的替代类，但比 Socket 和 ServerSocket 具有更多的功能，它支持非阻塞通信、可选择通信、异步通信和套接字对等通信等。

由于网络接收和传送数据均比较缓慢，常常导致线程阻塞，影响系统性能。在 Java 非阻塞通信模式中使用 Buffer 来缓冲数据。SocketChannel 和 ServerSocketChannel 对象一端连接着缓冲区，另一端连接着网络。

ServerSocketChannel 和 SocketChannel 在 Selector 中注册连接就绪事件、读就绪事件和写就绪事件。注册时创建一个 SelectionKey 对象，这个 SelectionKey 对象用来跟踪注册事件的句柄，其中包含有注册该对象的通道和缓冲区等信息。Selector 会一直监控已经注册的事件。当通道中有数据需要读取时，Selector 中就有已注册的读就绪事件发生，这时调用相关的程序可将通道中的数据读到缓冲区中，然后再提供给程序进行处理。同样地，当有数据需要通过网络发送时，数据先进入缓冲区，这时 Selector 中就发生写就绪事件，程序再利用通道向网络发送数据[1]。

非阻塞通信模式处理流程如下：

```
while(检测 Selector 对象，等待连接就绪事件、读就绪事
                    件或写就绪事件发生) {    //阻塞
    if(有客户连接)                          //非阻塞
        接收客户连接，创建 SocketChannel 对象与客
                                    户通信；
    if(某个 SocketChannel 输入流中有可读数据)
                                    //非阻塞
        从输入流读数据；
    if(某个 SocketChannel 输出流中有可写数据)
                                    //非阻塞
        向输出流写数据；
}
```

上述处理流程采用轮询工作方式，当某一种操作就绪时，执行该操作，否则就查看是否还有其他就绪操作可以执行，线程不会因为某一个连接、读、写等操作还没有就绪，就进入阻塞状态。在非阻塞通信模式中，只需要一个线程管理 Selector 对象就可以了，而不需要多线程。并且只有检测 Selector 对象时才有可能导致线程阻塞，因此可以大大提高系统的性能。

5 非阻塞通信在乡村可视化远程医疗系统中的应用研究

在乡村可视化远程医疗系统中，服务器端需要和各个客户端建立连接、接受客户端的操作请求并将操作结果返回给客户端。服务器只需要使用一个线程就可以处理客户端连接请求和向各个客户端读写数据，服务器自身进行的数据库操作可以使用另外的线程。

专家端和医务端在服务器端的数据库中进行查询、插入和修改等操作，这些操作信息是各种 SQL 语句，服务器返回给专家端和医务端的是查询、保存或修改结

果，它们可能是一个字符串、一条记录，也可能是多条记录。为了方便操作，这些信息均封装成 XML 格式，根标志为<teleMedi></teleMedi>。这种格式封装数据便于检验数据的完整性与正确性，也容易解析。在非阻塞通信中，除 boolean 类型以外，每种基本数据类型都有对应的缓冲区类，在本系统中使用 ByteBuffer 类对象数据缓冲区。程序中需要向网络发送数据时，数据进入缓冲区前先要把字符数据编码成字节数据，然后再由通道向网络发送，而从网络上接收的字节流数据先存放进缓冲区，解码后再由程序处理。

在非阻塞通信模式中，一次完整的数据发送可能需要多次读缓冲区，这避免了阻塞通信模式中读取数据时间拖得过长而导致的线程阻塞。每次读缓冲区后都要进行数据完整性检查，由于采用 XML 格式封装数据，所以很容易检验是不是读取到了完整的数据。

乡村可视化远程医疗系统中每次通过网络读写的数据长短不同。医疗中需要传送的数据最常见的长度小于 1 KB，因此缓冲区最初设定为 1 KB，当数据超过这个长度时，将缓冲区的容量扩大一倍，如果超过 2 KB，再将缓冲区扩大一倍，如此类推。使用这种可伸缩的方式可以更有效地利用内存资源。

6 非阻塞通信在乡村可视化远程医疗系统中的实现[2]

在非阻塞模式下，服务器主程序 MedicalSever 只使用一个线程，使用 Selector 监控接收连接就绪事件、读就绪事件和写就绪事件。限于篇幅，本文仅介绍几段主要的代码。

（1）MedicalServer 类的构造方法负责启动服务器，把它绑定到一个本地端口，主要代码如下：

```
selector=Selector.open();          //创建一个 Selector 对象
ssChannel=ServerSocketChannel.open();
                                   //创建 ServerSocketChannel 对象
ssChannel.configureBlocking(false);
                   //设置 ServerSocketChannel 为阻塞工作模式
ssChannel.socket().bind(new InetSocketAddress(port));
                                   //绑定到本地端口
```

（2）MedicalSever 类的 serve()方法负责轮询 Selector，主要代码如下：

```
public void serve() throws IOException{
    ssChannel.register(selector,SelectorKey.OP_ACCEPT);
                          //在 Selector 中注册连接就绪事件
    while(selector.select()>0){
      Set readyKeys=selector.selectedKeys();
                          //获取 Selector 中就绪事件集合
      Iterator it=readyKeys.iterator();
      while(it.hasNext()){
        SelectorKey key=null;
        try{
          key=(SelectionKey)it.next();
                                   //获取某个就绪事件
          it.remove();             //获取就绪事件后删除
          if(key.isReadable()){处理连接就绪事件}
          if(key.isReadable()){处理读就绪事件}
          if(key.isWritable()){处理写就绪事件}
        }catch(IOException e){  }}}
```

（3）处理连接就绪事件方法中的主要代码如下：

```
ServerSocketChannel ssc=(ServerSocketChannel)key.channel();
SocketChannel sChannel=(SocketChannel)ssc.accept();
                                   //获取关系的 SocketChannel
socketChannel.configureBlocking(false);
                          //将 SocketChannel 设置为非阻塞模式
ByteBuffer buffer=ByteBuffer.allocate(1024); //分配缓冲区
socketChannel.register(selector,SelectorKey.OP_READ|
                SelectionKey.OP_WRITER,buffer);
                                   //注册读就绪和写就绪事件
```

（4）处理读就绪事件方法中的主要代码如下：

```
public void receive(SelectionKey key)throws IOException{
  ByteBuffer sBuffer=(ByteBuffer)key.attachment();
                          //获取与 SelectorKey 绑定的缓冲区
  SocketChannel sChannel=
        (SocketChannel)key.channel();//获得通道
  ByteBuffer buff=ByteBuffer.allocate(1024)
                                   //创建辅助缓冲区
  sChannel.read(buff); //从通道中读数据暂存到 buff 中
  sBuffer.put(buff); }  //把 buff 中的内容拷贝到缓冲区
```

在非阻塞模式下，sChannel.read(buff)方法无法保证一次把全部数据都读完，因此只好把每次读到的数据先存放到 buff 中，并判断每次读到的数据是否以</teleMedi>结尾，当数据读取完整了，再复制到 sBuffer 中提交给程序。

（5）处理写就绪事件方法中的主要代码如下：

```
public void send(SelectionKey key)throws IOExceptio{
  ByteBuffer sBuffer=(ByteBuffer)key.attachment();
                          //获取需要发送数据的缓冲区
  SocketChannel sChannel=(SocketChannel)key.channel();
  synchronized(sBuffer)
  {    sBuffer.flip();     //将缓冲当前的位置值设为
       极限，将位置设为 0，为下一步复制数据做准备
       sChannel.write(sBuffer); //发送缓冲区中的数据
       sChannel.compact();}     //删除已发送的数据
```

（6）当每个缓冲区容量不够时，需要扩大缓冲区，主要代码如下：

```
protected void resizeBuffer(ByteBuffer bb)
{ if(bb.remaining()<10){           //判断剩余容量是否
                                   小于 10 B
    ByteBuffer bBuffer=ByteBuffer.allocate(bb.capacity()*2);
```

```
                                        //容量扩大一倍
    bb.flip();
    bBuffer.put(bb);          //把原缓冲区中数据复制到
                                        新缓冲区中
    bb=bBuffer;}}
```

乡村可视化远程医疗系统中使用Java非阻塞I/O通信技术，可以只使用一个主线程就能实现网络连接、读和写操作，避免了多线程中读或写引起的线程阻塞，大幅降低了服务器应用程序的开销，有效地提高了系统的性能。

参考文献

[1] 吴易，王凌.Java技术在P2P环境下的应用[J].微计算机信息，2005(3)：154-155.

[2] Java™ Platform Standard Edition 6 API Specification[S]. http://download.oracle.com/javase/6/docs/api/.

(收稿日期：2010-10-31)

作者简介：

顾和明，男，1968年生，硕士研究生，主要研究方向：网络技术、流媒体技术。