1.Netty学习前需要掌握知识

-- 1.牢固的JAVA基础

-- 2.熟悉Linux服务器

-- 3.有基础的网络知识

2.异步事件驱动NIO框架Netty介绍

-- 1.Netty是由JBOSS提供的一个java开源框架,是业界最流行的NIO框架，整合了多种协议(包括FTP,SMTP,HTTP等各种二进制文本协议)的实现经验，

API使用简单，框架成熟稳定，社区活跃，互联网、大数据、网络游戏、电信通信行业进过验证。

-- 2.使用Netty框架的产品

--- 1.搜索引擎框架 elasticsearch

--- 2.Hadopp子项目Avro项目,使用Netty作为底层通信框架

--- 3.阿里巴巴开源的RPC框架 Dubbo

3.Netty开发环境

-- 1.IDEA + JDK8 + Netty4.x

4.使用JDK自带BIO编写client-server通信

-- 1.BIO模式:同步阻塞，传统Socket编程，一客户端一线程

-- 2.Server核心代码:

ServerSocket server =new ServerSocket(port);

Socket socket=null;

while(true){

socket=server,accept();

new Thread(new xxxHandler(socket)).start();

}

-- 3.client核心代码:

Socket socket=new Socket(ip,port);

socket.getInputStream();

socket.getOutputStream();

5.BIO编写client-server通信优缺点分析

-- 1.优点:模型简单，编码简单

-- 2.缺点:性能瓶颈，请求数和线程数 N:N关系，高并发情况下，cpu切换线程上下文损耗大

-- 3.案例：tomcat7之前，都是使用BIO,7之后使用NIO

-- 4.改进:伪NIO,使用线程池去处理业务逻辑

6.通俗概念讲解 同步异步、阻塞和非阻塞

-- 1.同步阻塞:你把衣服丢到洗衣机，然后看着洗衣机洗完，洗好后再去晾衣服（你就干等着，啥都不做，阻塞在那边）

-- 2.同步非阻塞:你把衣服丢到洗衣机洗，然后回客厅去做其他事情，定时去阳台看洗衣机是不是洗完了，洗好后再去晾衣服（等待期间可做其他事情）

-- 3.异步阻塞:几乎没有这个情况，可以忽略，异步应该就非阻塞

-- 4.异步非阻塞:你把衣服丢到洗衣机，然后回客厅做其他事情，衣服洗好后会自动去晾衣服，晾完衣服后放个音乐通知你洗好衣服并且衣服晾好了

7.Linux网络编程中的五种I/O模型理解

网络IO,用户程序和内核的交互为基础进行理解

IO操作分为两步：发起IO请求等待数据准备，实际IO操作(洗衣服，晾衣服)

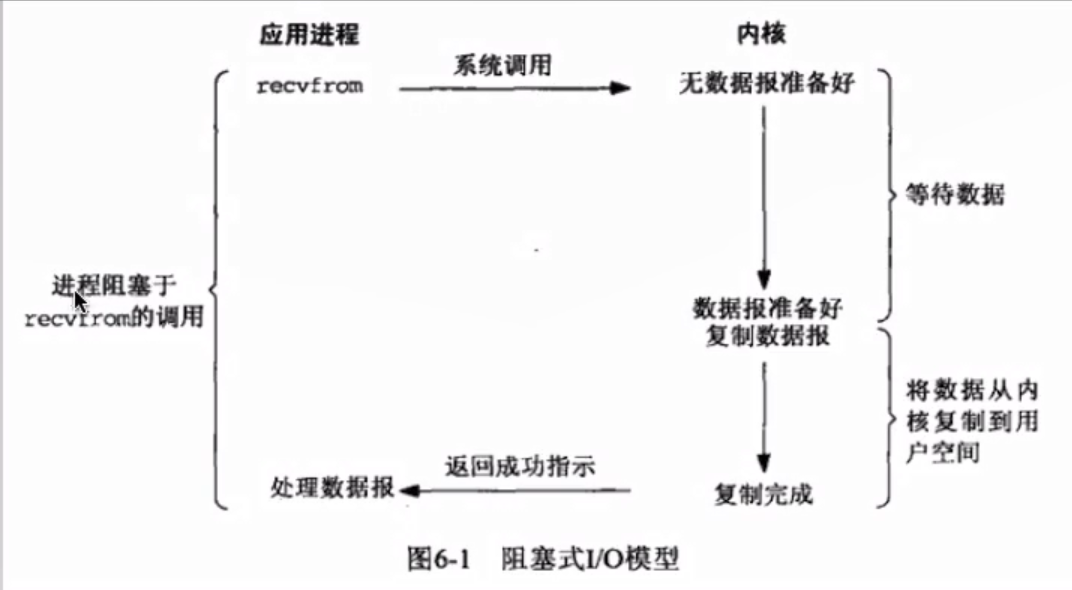
同步需要主动读写数据，在读写数据的过程中还是会阻塞（好比晾衣服阻塞了）

异步仅仅需要IO操作完毕的通知，并不主动读写数据，由操作系统内核完成数据的读写（机器人帮你自动晾衣服）

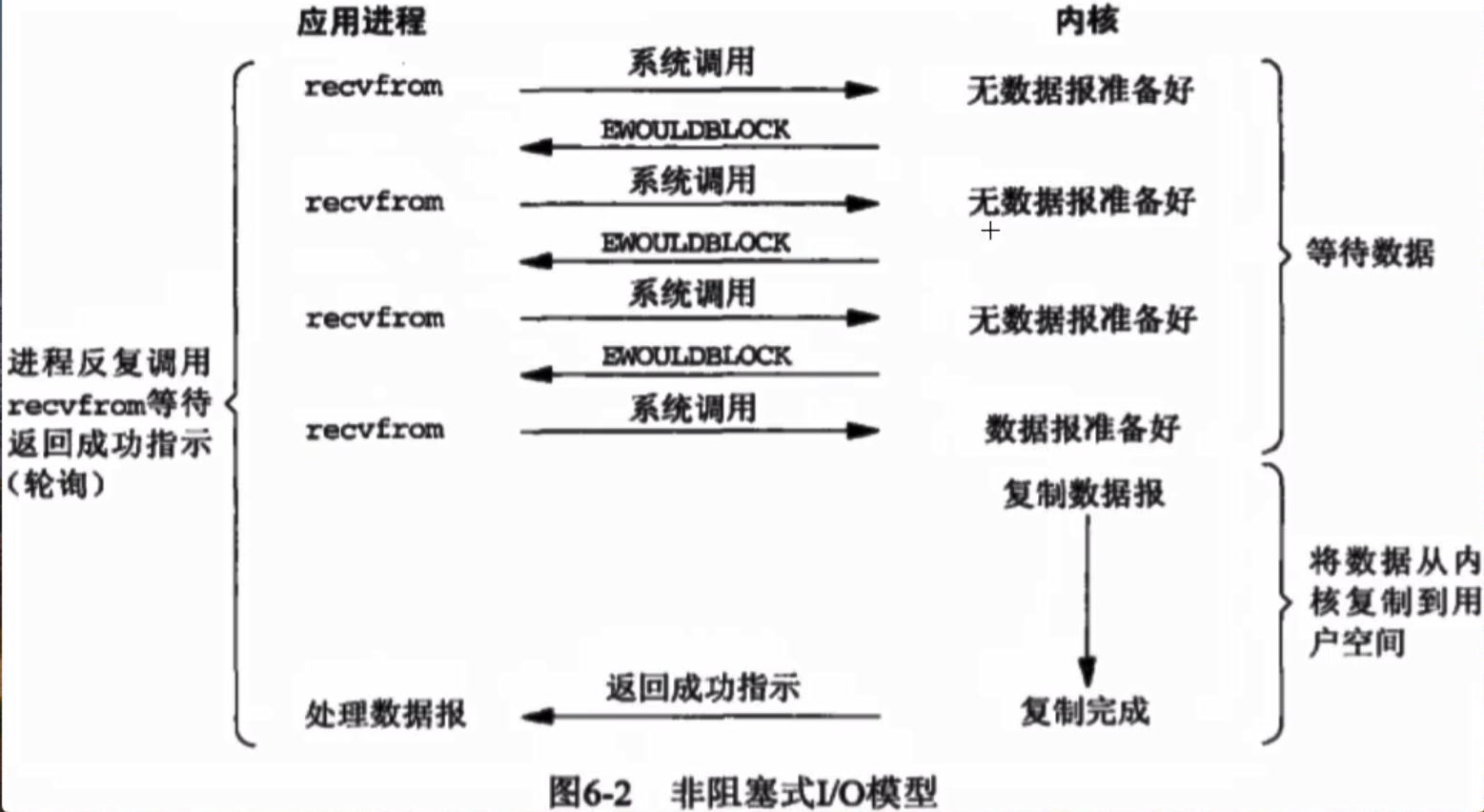
五种IO模型:阻塞IO,非阻塞IO,多路复用IO,信号驱动IO和异步IO

前四种都是同步IO，在内核数据copy到用户控件都是阻塞的

--- 1. 阻塞式IO



--- 2.非阻塞IO



--- 3.IO多路复用(select,poll,epoll……..)

一个线程去处理多个请求

IO多路复用是阻塞在select,epoll这样的系统调用，没有阻塞在真正的IO系统调用如

Recvfrom,进程受阻于select调用，等待可能多个套接口中的任何一个变为可读。

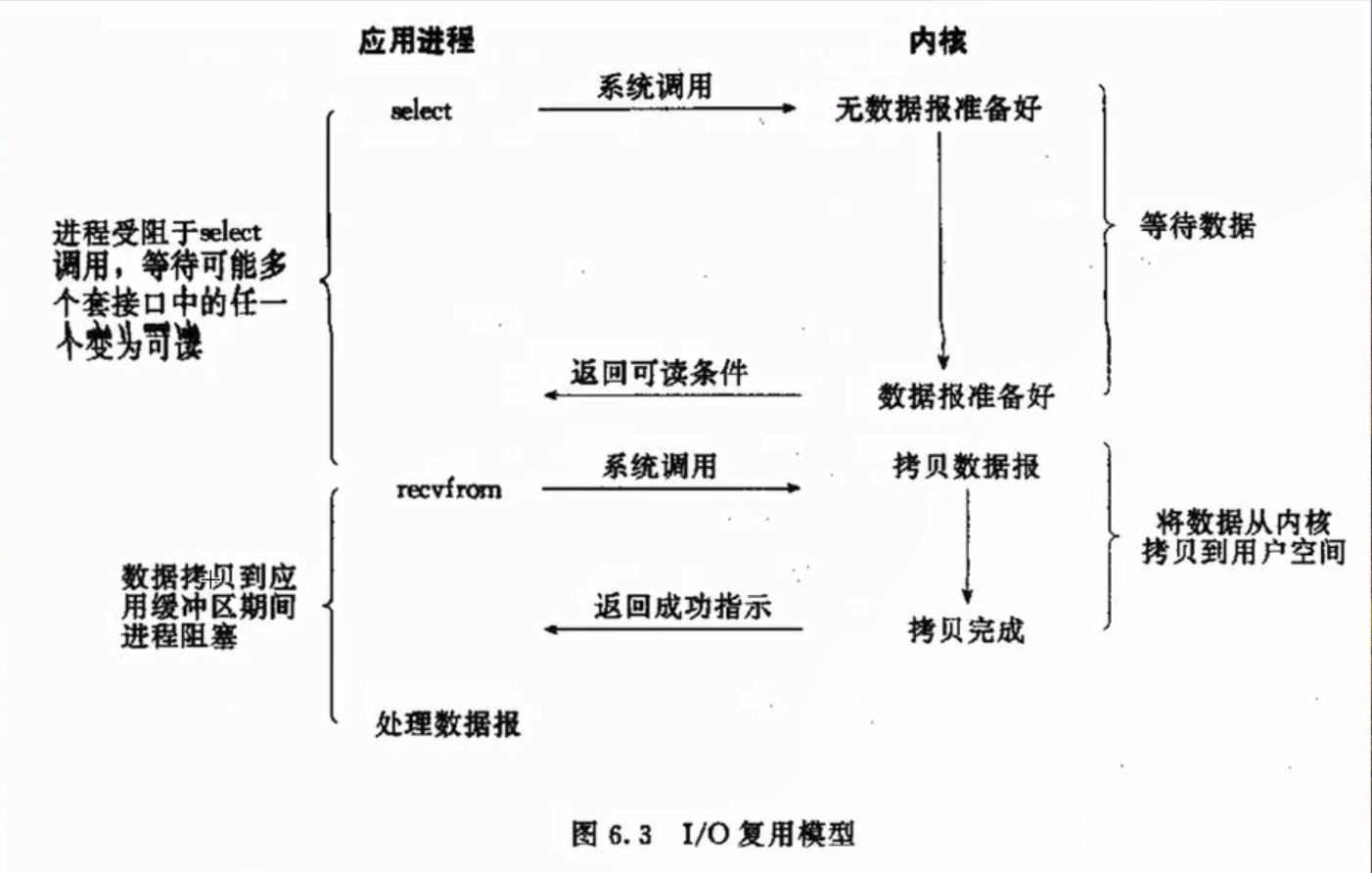
IO多路复用使用两个系统调用(select和recvfrom)

阻塞IO只调用了一个系统调用(recvfrom)

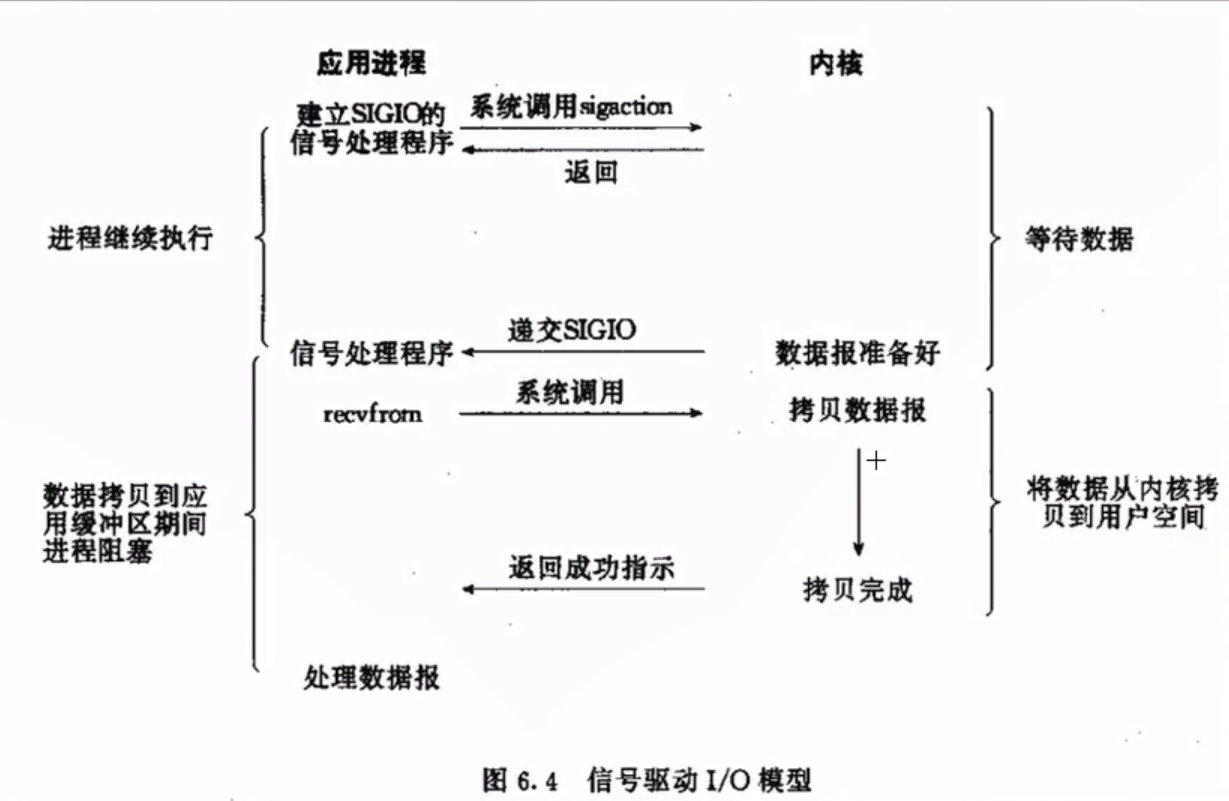
Select/epoll的核心是可以同时处理多个connection,而不是更快，所以连接数不高的话，性能不一定比多线程+阻塞IO好

多路复用模型中，每一个socket,设置为non-blocking,阻塞是select函数block，而不是

Socket阻塞的

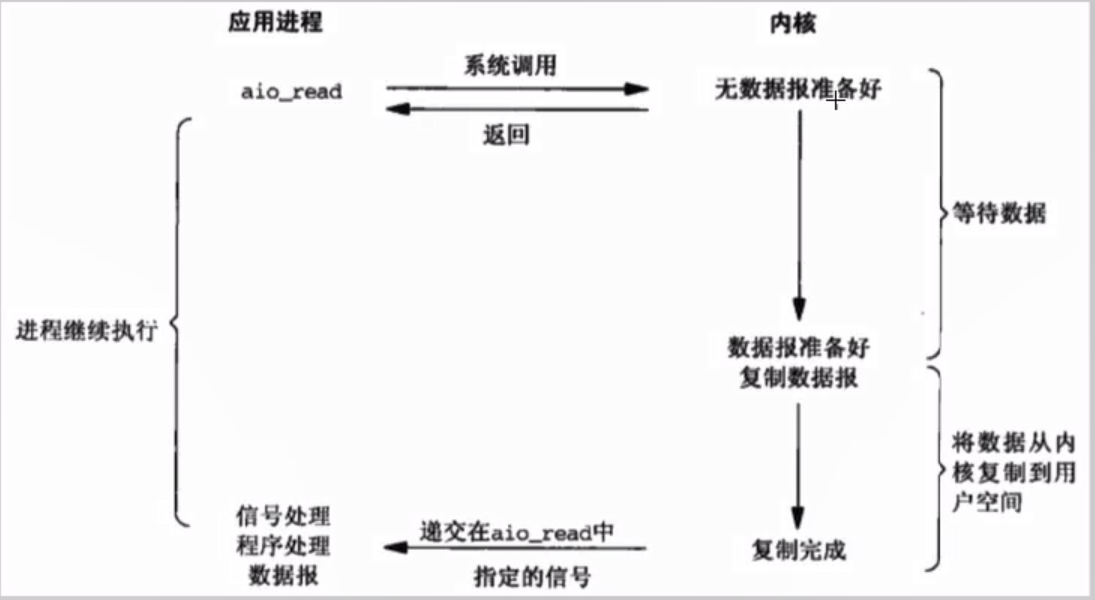


--- 4.信号驱动IO



--- 5.异步IO

Future-listener机制



总结点：

--IO操作分为两步

-----1.发起IO请求，等待数据准备(waiting for the data to be ready)

------2.实际的IO操作，将数据从内核拷贝到进程中(copying the data from the kernel to

the process)

前四种IO模型都是同步IO操作，他们的第二阶段是一样的:数据从内核复制到应用缓

冲期间（用户空间），区别在于第一阶段，进程阻塞于recvfrom调用或者select()函数。

异步IO在这两个阶段都要处理。

阻塞IO和非阻塞IO的区别在于第一步，发起IO请求是否会被阻塞，如果阻塞直到完

成那么就是传统的阻塞IO,如果不阻塞就是非阻塞IO

同步IO和异步IO的区别在于第二步骤是否阻塞，如果实际的IO读写阻塞请求进程，

那么就是同步IO,如果不阻塞，而是操作系统帮你完成IO操作再将结果返回给你，那

么就是异步IO

核心点:

阻塞和非阻塞指的是线程的状态，异步和同步说的是消息的通知机制

同步需要主动读写数据，异步不需要

同步IO和异步IO是针对用户应用程序和内核的交互