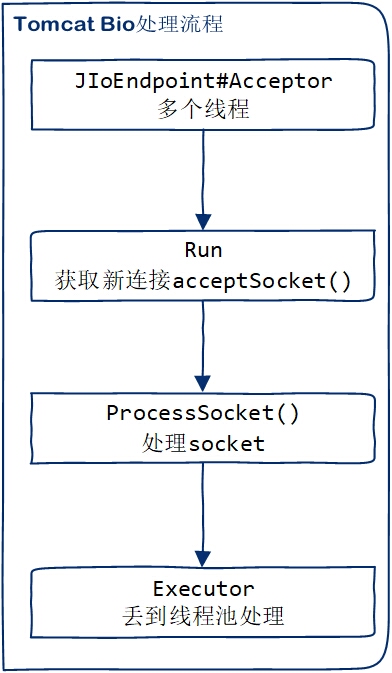
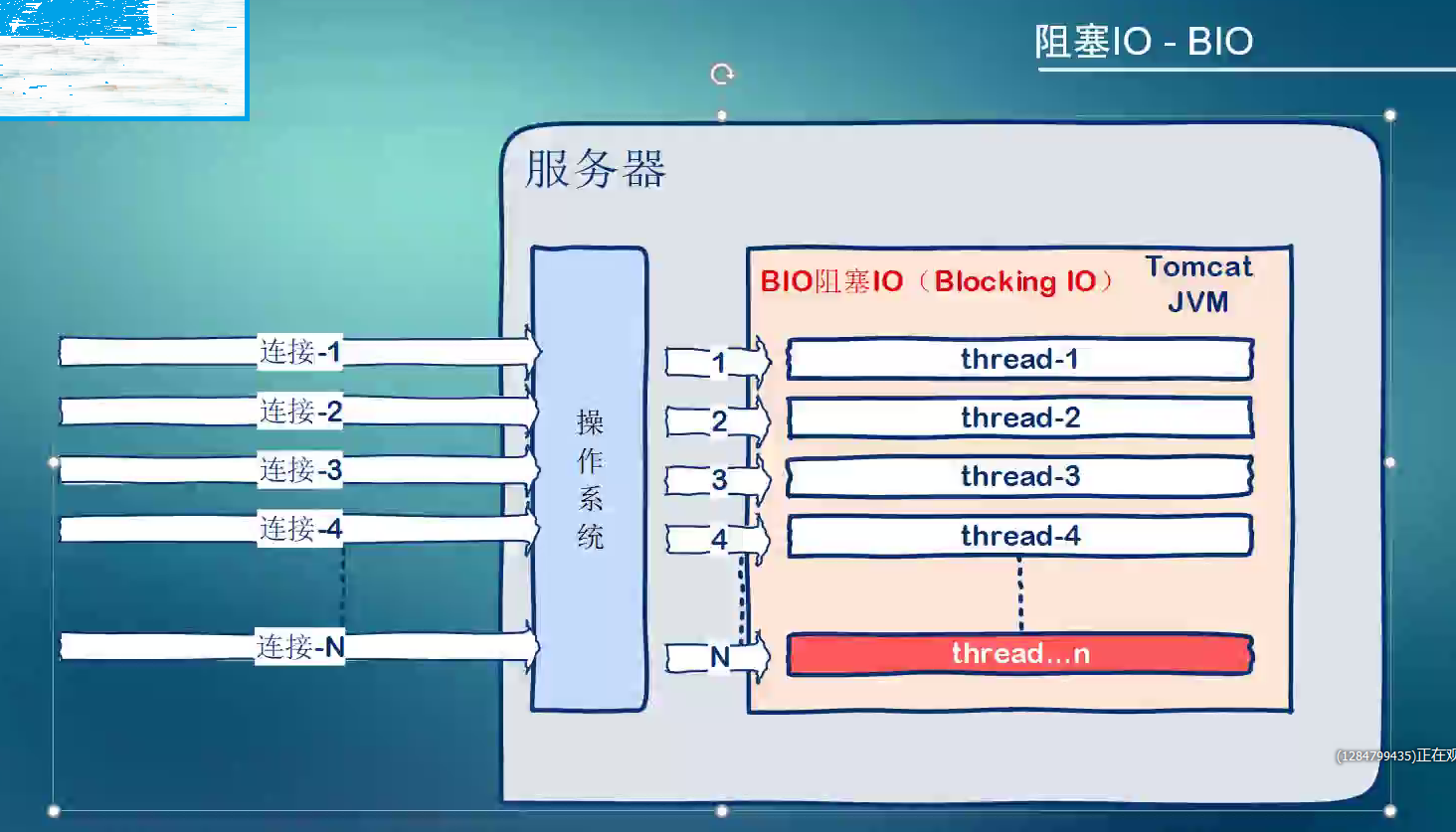
*tomcat服务器探索解析*

*<!--bio 阻塞io: blocking io*

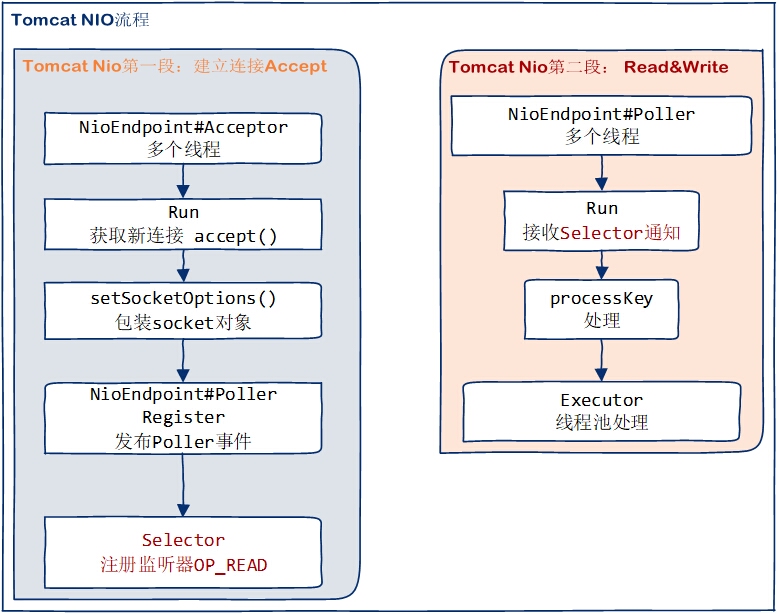
*Socket socket=socketServer.accept();*



*根据bio的处理流程来看，显然有多少个客户端连接，服务器的线程池就会对应开启多少个线程处理连接，而每一个连接都在占用着连接的资源。*



*nio非阻塞io: no blocking io*



*在tomcat8.5之前默认使用的是bio；nio被隐藏，可以根据实际业务需要定制nio；而在tomcat8.5之后则改为了默认nio的服务连接模式。*

*nio模式的处理流程分为两个阶段：*

*第一阶段：*

*1.接受连接；2.处理连接（注册轮询监控和数据监听）；*

*同样是一个线程对应处理一个连接，为每个连接注册为监控选择器selector的对象，为每一个连接（线程）注册一个监听器OP\_READ。*

*第二阶段：*

*轮询监控对每个线程进行轮询，在轮询时接受OP\_READ数据监听器的事件通知；如果连接中包含数据交互的事件，则该连接需要被通知然后交由线程池处理，如果有数据传输就建立连接，如果没有就继续定时监听连接状态。  
 netty也是基于nio实现的-->*

tomcat服务器nio原理示例：

public class MyNioHttpServer{

//不是ServerSocket

//nio中的channel

socketChannel = ServerSocketChannel.open();

socketChannel.configureBlocking(false);

socketChannel.bind(new InetSocketAddress(port));

System.out.println("NIO启动-->端口:"+port);

//获取一个选择器

//NIO 中的selector

// 通俗来说：由操作系统提供事件通知机制

MyNioHttpServer.selector = Selector.open();

//等级:表示对这个通道上的OP\_ACCEPT事件感兴趣，并且返回一个标记

//此处表示，希望收到socket通道8080端口上建立连接这个通知

//告诉selector.如果我这个端口有链接建立了，就告知一下

socketChannel.register(MyNioHttpServer.selector ,SelectionKey.OP\_ACCEPT);

while(true){

//pull

//如果没有新的socket与服务器有链接或者是数据交互，这里就会等它1秒内是不是有连接//到来

MyNioHttpServer.selector.select(1000);

//拉取事件

Set<SelectionKey> selected = MyNioHttpServer.selector.selectKeys();

Iterator<SelectionKey> iter = selected.iterator();

while(iter.hasNext()){

//获取注册在上面的标记

SelectionKey key = iter.next();

if(key.isAcceptable()){

//是否是新连接，判断是否OP\_ACCEPT的通知

//处理连接

System.out.println("有新的连接，当前线程数量：“ + MyNioHttpServer.threadPoolExcutor.getActiveCount());

//获取新的连接对象

SocketChannel chan = socketChannel.accept();

chan.configureBlocking(false);

//注册一个监听。继续告诉selector，这个新的连接你帮我监控着。

//有数据有数据了再通知我。希望收到该连接上OP\_READ"数据读取"事件的通知。

chan.register(MyNioHttpServer.selector , SelectionKey.OP\_READ);

}else if(key.isReadable()){

//说明此时OP\_READ已经监听到该连接有数据传输

//拿到连接

SocketChannel socketChannel = (SocketChannel) key.channel();

//告诉selector，这个socket连接正在处理中，不要再通知我

key.cancel();

socketChannel.configureBlocking(false);

//把有数据的连接交给线程池区处理

MyNioHttpServer。threadPoolExcutor.excute(new NioSocketProcessor(socketChannel));

}

//取出后删除该链接

iter.remove();

}

selected.clear();

//过掉重置cancelled keys

MyNioHttpServer.selector.selectNow();

}

}

}