## Netty

Netty是基于java NIO(NoneBlocking IO)技术封装的一套框架

Netty是由JBOSS提供的一个java开源框架。Netty提供异步的、事件驱动的网络应用程序框架和工具，用以快速开发高性能、高可靠性的网络服务器和客户端程序。 也就是说，Netty 是一个基于NIO的客户，服务器端编程框架，使用Netty 可以确保你快速和简单的开发出一个网络应用，例如实现了某种协议的客户，服务端应用。Netty相当简化和流线化了网络应用的编程开发过程，例如，TCP和UDP的socket服务开发。 “快 速”和“简单”并不意味着会让你的最终应用产生维护性或性能上的问题。Netty 是一个吸收了多种协议的实现经验，这些协议包括FTP,SMTP,HTTP，各种二进制，文本协议，并经过相当精心设计的项目，最终，Netty 成功的找到了一种方式，在保证易于开发的同时还保证了其应用的性能，稳定性和伸缩性。

如果没有Netty？

远古：

java.net + java.io

近代：

Java.nio

小结

1）本质：JBoss做的一个Jar包

2）目的：快速开发高性能、高可靠性的网络服务器和客户端程序

3）优点：提供异步的、事件驱动的网络应用程序框架和工具

通俗的说：一个好使的处理Socket的东东

## 异步和同步

同步和异步是针对应用程序和内核的交互而言的，同步指的是用户进程触发 IO 操作并

等待或者轮询的去查看 IO 操作是否就绪，而异步是指用户进程触发 IO 操作以后便开始做自己的事情，而当 IO 操作已经完成的时候会得到 IO 完成的通知。

## 为什么netty基于NIO而不是AIO?

AIO是异步非阻塞的，NIO是同步非阻塞的。理论上讲，AIO的吞吐量肯定比NIO的要大。两种IO的概念如下：

NIO模型: 同步非阻塞。服务器实现模式为一个请求一个线程，但客户端发送的连接请求都会注册到多路复用器上，多路复用器轮询到连接有I/O请求时才启动一个线程进行处理。

AIO模型: 异步非阻塞。服务器实现模式为一个有效请求一个线程，客户端的I/O请求都是由OS先完成了再通知服务器应用去启动线程进行处理。注：AIO又称为NIO2.0，在JDK7才 开始支持。

主要原因如下：

Netty不看重Windows上的使用，在Linux系统上，AIO的底层实现仍使用EPOLL，没有很好实现AIO，因此在性能上没有明显的优势，而且被JDK封装了一层不容易深度优化。

Netty整体架构是reactor模型, 而AIO是proactor模型, 混合在一起会非常混乱,把AIO也改造成reactor模型看起来是把epoll绕个弯又绕回来。

AIO还有个缺点是接收数据需要预先分配缓存, 而不是NIO那种需要接收时才需要分配缓存, 所以对连接数量非常大但流量小的情况, 内存浪费很多。

Linux上AIO不够成熟，处理回调结果速度跟不上处理需求，比如外卖员太少，顾客太多，供不应求，造成处理速度有瓶颈（待验证）。

作者原话引用：

作者原话：

Not faster than NIO (epoll) on unix systems (which is true)

There is no daragram suppport

Unnecessary threading model (too much abstraction without usage)

在unix系统上不比NIO（epoll）快（这是真的）

没有daragram支持

不必要的线程模型（没有使用过多的抽象）

## Netty线程模型

高度可定制的线程模型：单线程模型（监听线程组与客户端任务处理多线程租为同一个，并且数量为1）、多线程模型（客户端请求监听线程组数量为1，客户端任务处理多线程组为多线程）、主从多线程模型（两者皆为多线程）

## Netty现实应用

随着网站规模的不断扩大，系统并发访问量也越来越高，传统基于tomcat等web容器垂直架构已经无法满足需求，需要拆分应用进行服务化，以提高开发和维护效率，从组网情况看，垂直的架构拆分之后，系统采用分布式部署，各节点之间需要远程服务调用，高性能的rpc框架必不可少。netty作为异步高性能的通信框架，往往作为基础通信组件被这些rpc框架使用。

典型的应用有：阿里分布式服务框架 Dubbo 的 RPC 框架使用 Dubbo 协议进行节点间通信，Dubbo 协议默认使用 Netty 作为基础通信组件，用于实现各进程节点之间的内部通信。

除了 Dubbo 之外，淘宝的消息中间件 RocketMQ 的消息生产者和消息消费者之间，也采用 Netty 进行高性能、异步通信。

除了阿里系和淘宝系之外，很多其它的大型互联网公司或者电商内部也已经大量使用 Netty 构建高性能、分布式的网络服务器。

https://www.zhihu.com/search?type=content&q=netty

## IO多路复用

其实I/O多路复用这个坑爹翻译可能是这个概念在中文里面如此难理解的原因。所谓的I/O多路复用在英文中其实叫 I/O multiplexing.重要的事情再说一遍： I/O multiplexing 这里面的 multiplexing 指的其实是在单个线程通过记录跟踪每一个Sock(I/O流)的状态来同时管理多个I/O流. 发明它的原因，是尽量多的提高服务器的吞吐能力。