Netty线程模型

Netty简介

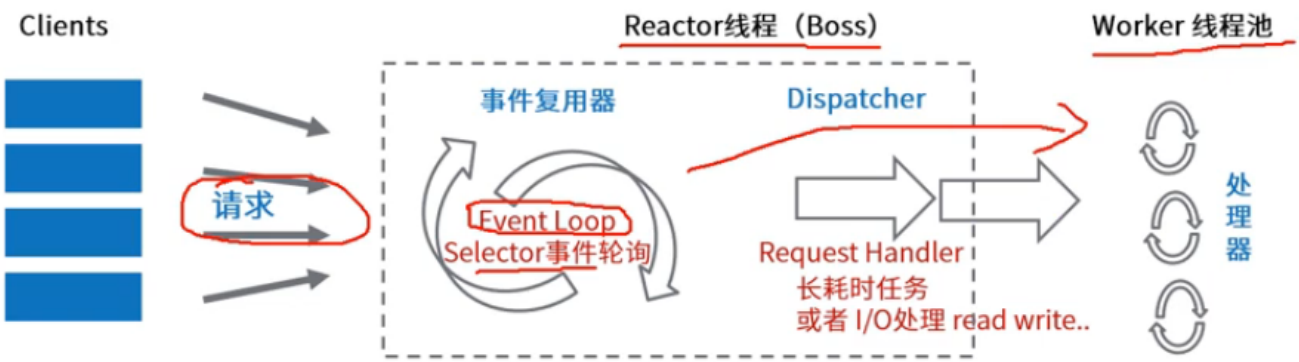
Netty时一个高性能，高可扩展的异步事件驱动的网络应用程序框架，它极大地简化了TCP和UDP客户端和服务器开发等网络编程。

1. Reactor线程模型：一种高性能的多线程程序设计思路。
2. Netty中自己定义的Channel概念：增强版的通道概念。
3. ChannelPipeline职责链设计模式：事件处理机制。
4. 内存管理：增强的ByteBuf缓冲区。

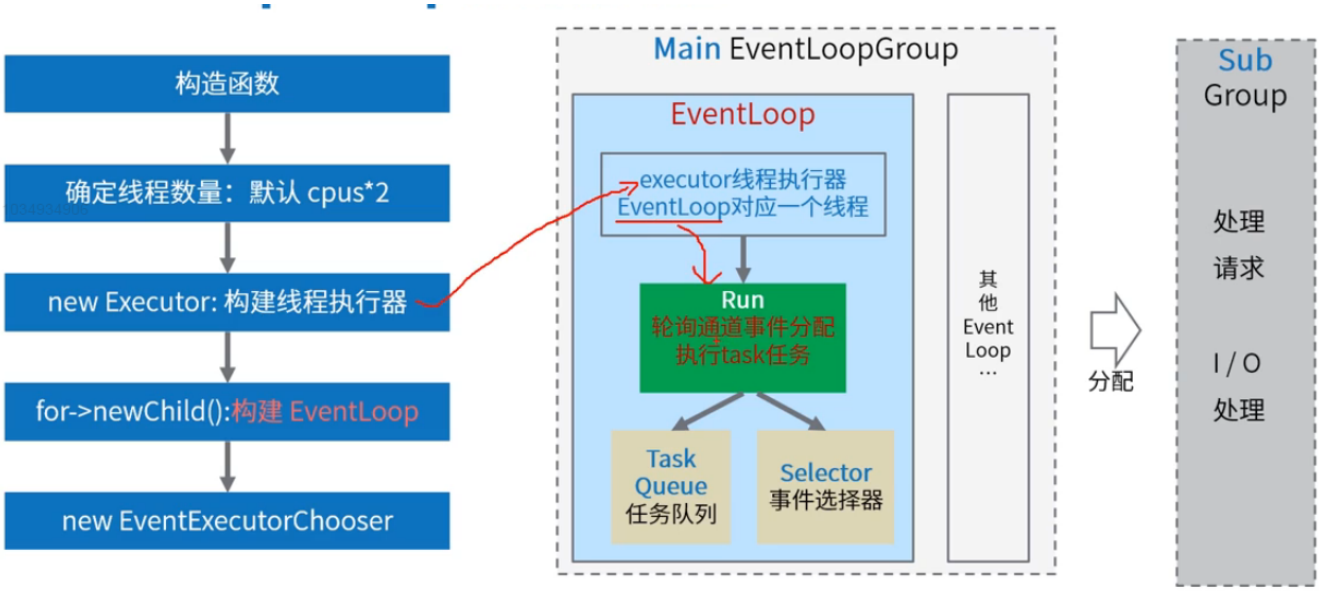
Netty线程模型

为了让NIO处理更好的利用多线程特性，Netty实现了Reactor线程模型。Reactor模型中有四个核心概念：

1. Resources资源(请求/任务)。
2. Synchronouts Event Demultiplexer同步事件复用器。
3. Dispatcher分配器。
4. Request Handler请求处理器。



EventLoopGroup初始化过程



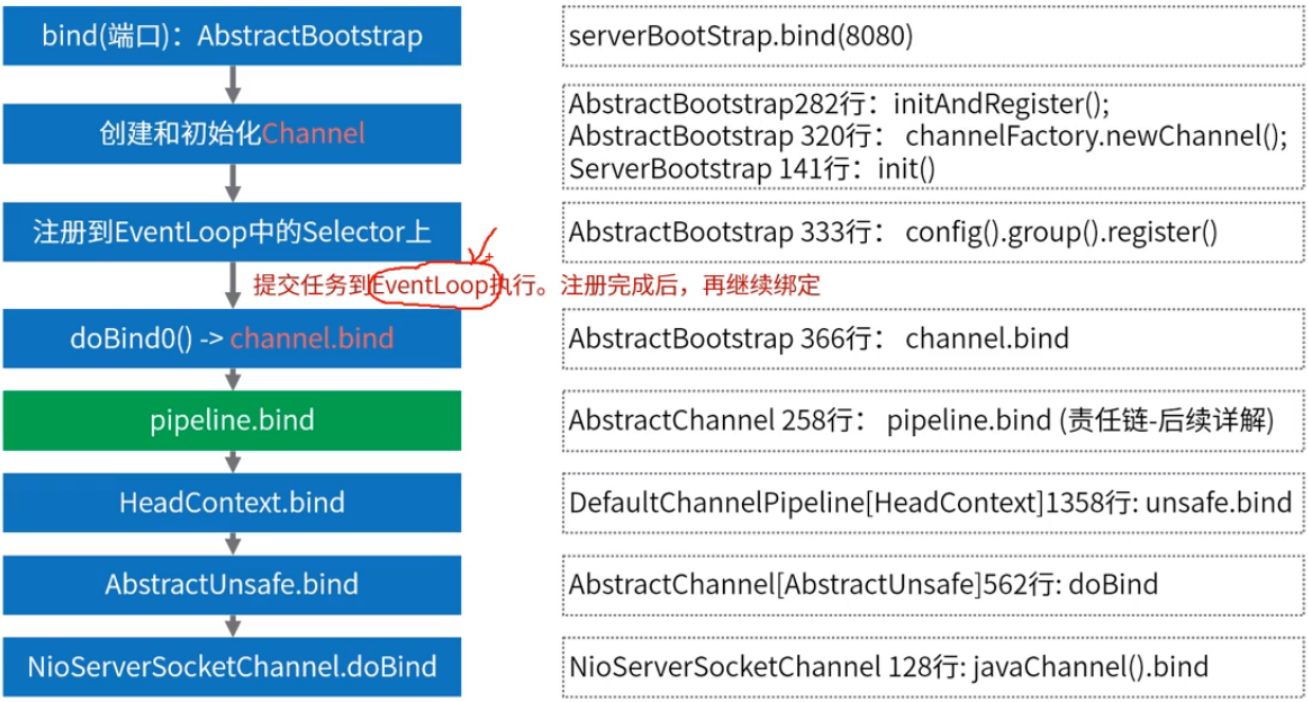
两组EventLoopGroup（Main&Sub）处理不同通道不同的事件

EventLoop的启动

EventLoop自身实现了Executor接口，当调用executor方法提交任务时，则判断是否启动，未启动则调用内置的executor创建新线程来触发run方法执行。

executor：请求执行任务	SingleThreadEventExecutor 772行
addTask：增加到任务队列	SingleThreadEventExecutor 778行
判断是否是EventLoop自身调用	SingleThreadEventExecutor 779行
startThread -> doStartThread	SingleThreadEventExecutor 883行
使用executor创建新线程执行run	SingleThreadEventExecutor 898行，909行

Bind绑定端口过程



Channel概念

netty中的Channel是一个抽象的概念，可以理解为对JDK NIO Channel的增强和拓展。增加了很多属性和方法，完整信息可以看代码注释，下面罗列几个常见的属性和方法：

AbstractChannel
- pipeline DefaultChannelPipeline // 通道内事件处理链路 - eventLoop EventLoop // 绑定的EventLoop,用于执行操作 - unsafe Unsafe // 提供I/O相关操作的封装 …
config() ChannelConfig // 返回通道配置信息 + read() Channel // 开始读数据，触发读取链路调用 + write(Object msg) ChannelFuture // 写数据，触发链路调用 + bind(SocketAddress SocketAddress) ChannelFuture // 绑定 …