Netty 异步事件驱动NIO 与原生NIO对比

原生NIO 服务端 code

public class NIOServer {

//通道管理器

private Selector selector;

public void initServer(int port) throws IOException {

// 获得一个ServerSocket通道

ServerSocketChannel serverChannel = ServerSocketChannel.open();

// 设置通道为非阻塞

serverChannel.configureBlocking(false);

// 将该通道对应的ServerSocket绑定到port端口

serverChannel.socket().bind(new InetSocketAddress(port));

// 获得一个通道管理器

this.selector = Selector.open();

//将通道管理器和该通道绑定，并为该通道注册SelectionKey.OP\_ACCEPT事件,注册该事件后，

//当该事件到达时，selector.select()会返回，如果该事件没到达selector.select()会一直阻塞。

serverChannel.register(selector, SelectionKey.OP\_ACCEPT);

}

public static void main(String[] args) throws IOException {

NIOServer server = new NIOServer();

server.initServer(8000);

}

}

**Netty的NIO服务端**

// Configure the server.

EventLoopGroup bossGroup = new NioEventLoopGroup(1);

EventLoopGroup workerGroup = new NioEventLoopGroup();

try {

ServerBootstrap b = new ServerBootstrap();

b.group(bossGroup, workerGroup)

.channel(NioServerSocketChannel.class)

.option(ChannelOption.SO\_BACKLOG, 100)

.handler(new LoggingHandler(LogLevel.INFO))

.childHandler(new ChannelInitializer<SocketChannel>() {

@Override

public void initChannel(SocketChannel ch) throws Exception {

ChannelPipeline p = ch.pipeline();

p.addLast(new EchoServerHandler());

}

});

// Start the server.

ChannelFuture f = b.bind(PORT).sync();

// Wait until the server socket is closed.

f.channel().closeFuture().sync();

} finally {

// Shut down all event loops to terminate all threads.

bossGroup.shutdownGracefully();

workerGroup.shutdownGracefully();

}

作者：linking12  
链接：https://www.jianshu.com/p/d5680a2c4d12  
來源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。