Tech Talk

——An introduction to Netty

2015/7/5

Table of Contents

http://netty.io/

Definition

What is Netty?

Netty is an asynchronous event-driven network application framework for rapid development of maintainable high performance protocol servers & clients.

Netty一种用来快速开发高性能、高可靠性的网络服务器和客户端的异步的事件驱动型的网络应用框架。

简单的一句话概括：Netty是基于 JAVA NIO 类库的异步通信框架。

Usage

1、开发异步、非阻塞的 TCP 网络应用程序；

2、开发异步、非阻塞的 UDP 网络应用程序；

3、开发异步文件传输应用程序；

4、开发异步 HTTP 服务端和客户端应用程序；

5、提供对多种编解码框架的集成，包括谷歌的 Protobuf、Jbossmarshalling、Java 序列化、压缩编解码、XML 解码、字符串编解码等，可以非常方便的实现私有协议栈编解码框架的二次定制和开发；

6、IP 黑白名单控制；

7、打印消息码流；

8、流量控制和整形；

9、性能统计；

10、基于链路空闲事件检测的心跳检测

……

Usage

阿里分布式服务框架 Dubbo

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

淘宝消息中间件 RocketMQ

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。

移动App推送服务器

发送心跳包，维持TCP长连接

华为的异步高性能网关平台

从2011年开始，华为主要使用Netty进行通信软件的开发

用作大数据各节点间的通信。

Hadoop 的高性能通信和序列化组件 Avro 的 RPC 框架

默认采用 Netty 进行跨节点通信

Advantage

1) API使用简单，开发门槛低；

2) 功能强大，预置了多种编解码功能，支持多种主流协议；

3) 定制能力强，可以通过ChannelHandler对通信框架进行灵活地扩展；

4) 性能高，通过与其他业界主流的NIO框架对比，Netty的综合性能最优；

5) 成熟、稳定，Netty修复了已经发现的所有JDK NIO BUG，业务开发人员不需要再为NIO的BUG而烦恼；

6) 社区活跃，版本迭代周期短，发现的BUG可以被及时修复，同时，更多的新功能会加入；

7) 经历了大规模的商业应用考验，质量得到验证。在互联网、大数据、网络游戏、企业应用、电信软件等众多行业得到成功商用，证明了它已经完全能够满足不同行业的商业应用。

By李林锋，华为软件平台架构部架构师，有7年NIO设计和开发经验，精通Netty、Mina等NIO框架和平台中间件。

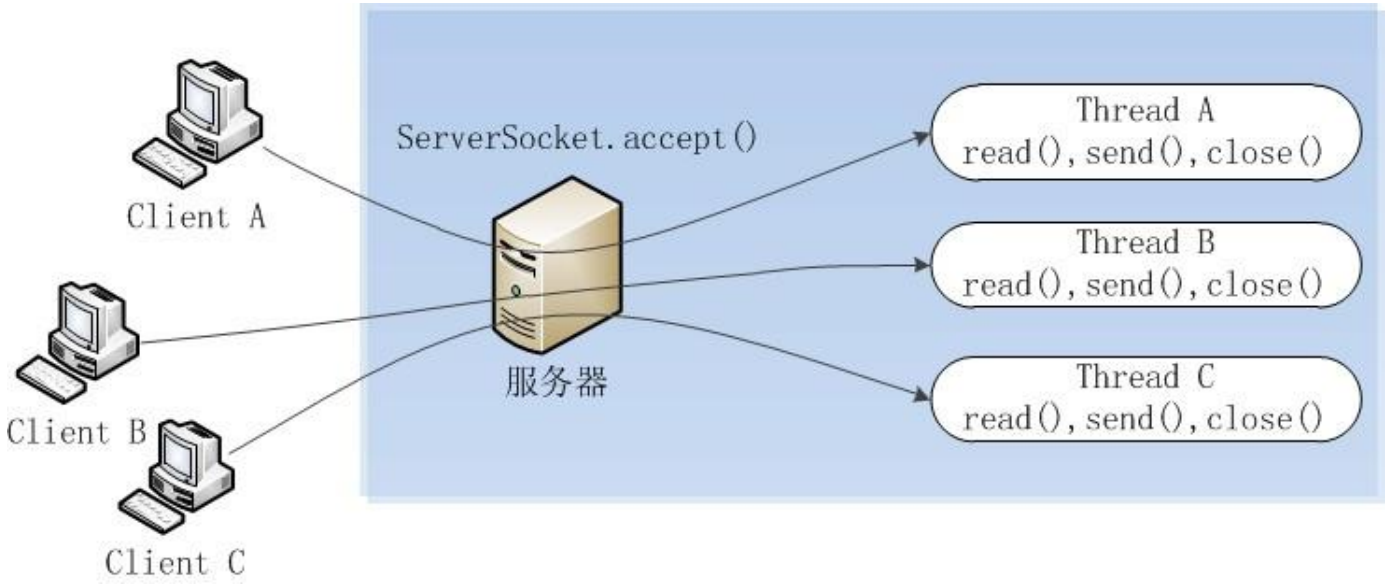
http://www.infoq.com/cn/articles/practice-of-java-nio-communication-framework?utm_source=tuicool

BIO VS NIO

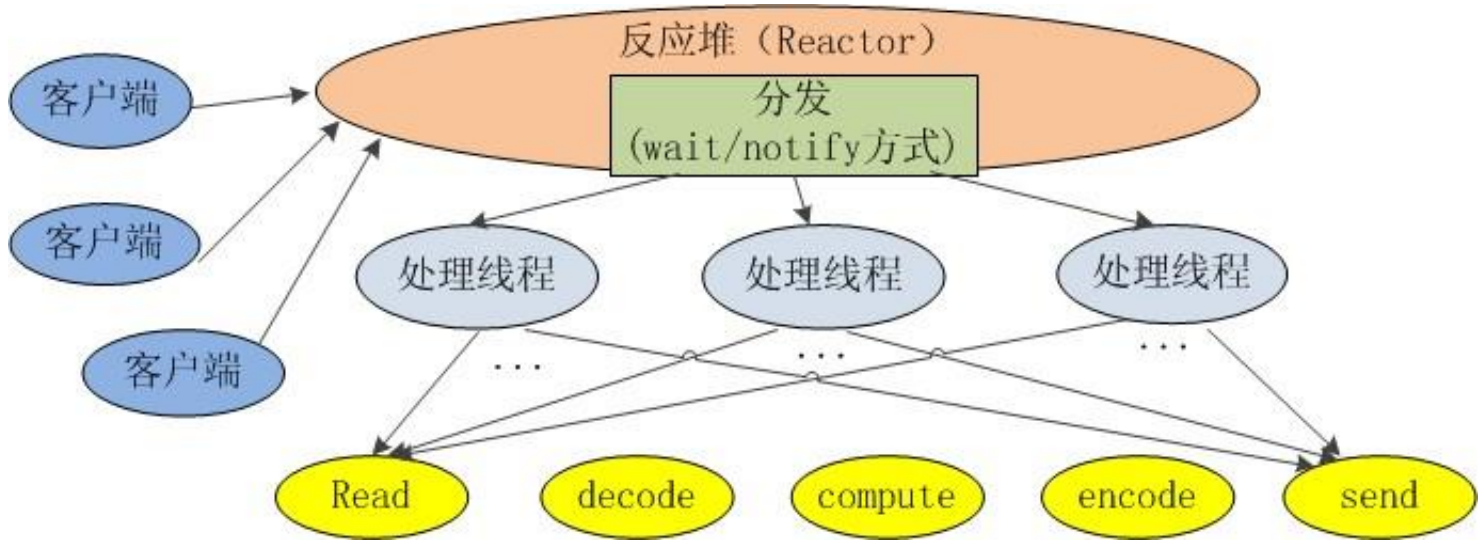
What does Netty look like?

传统的 Socket ，BIO

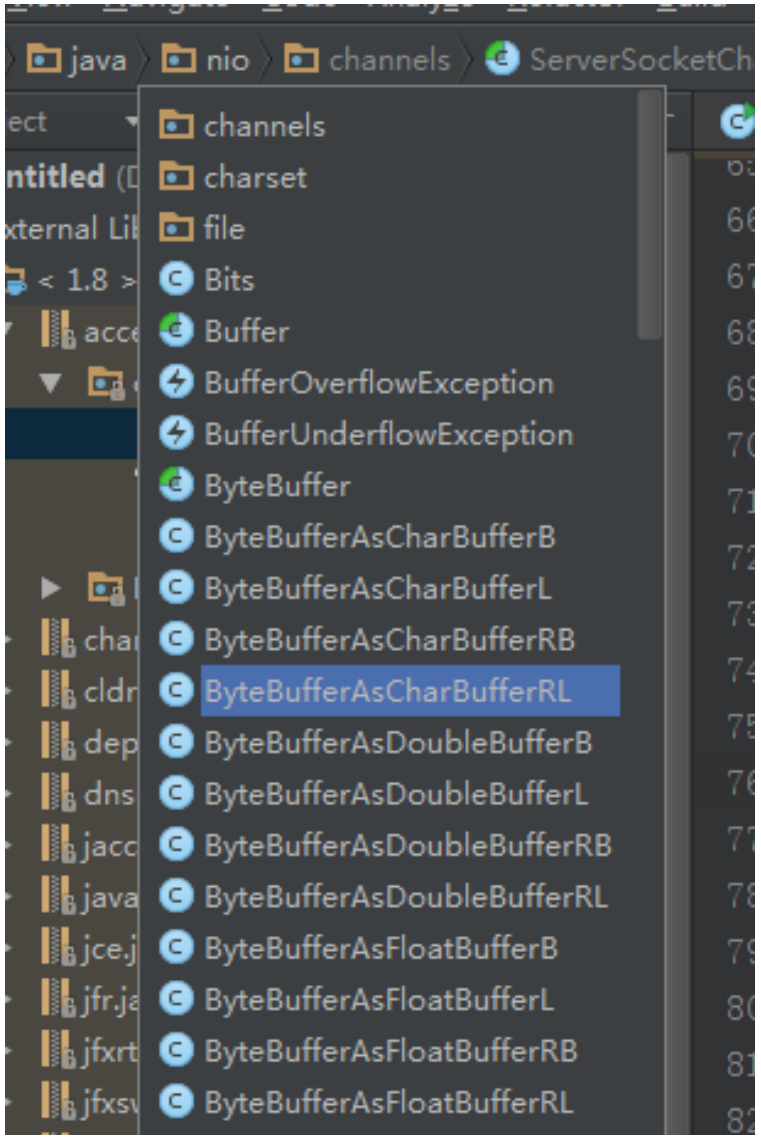
同步阻塞



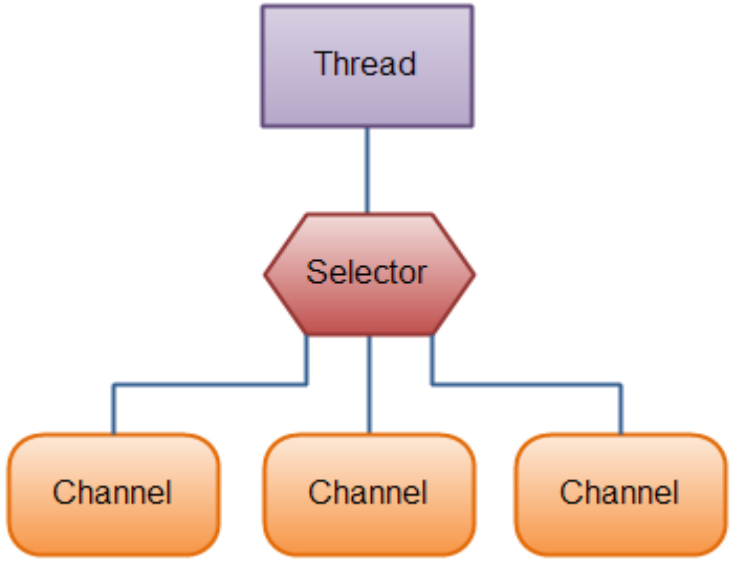
NIO,异步非阻塞

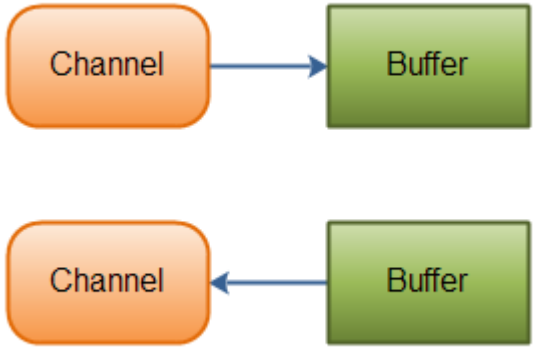


What does Netty look like?

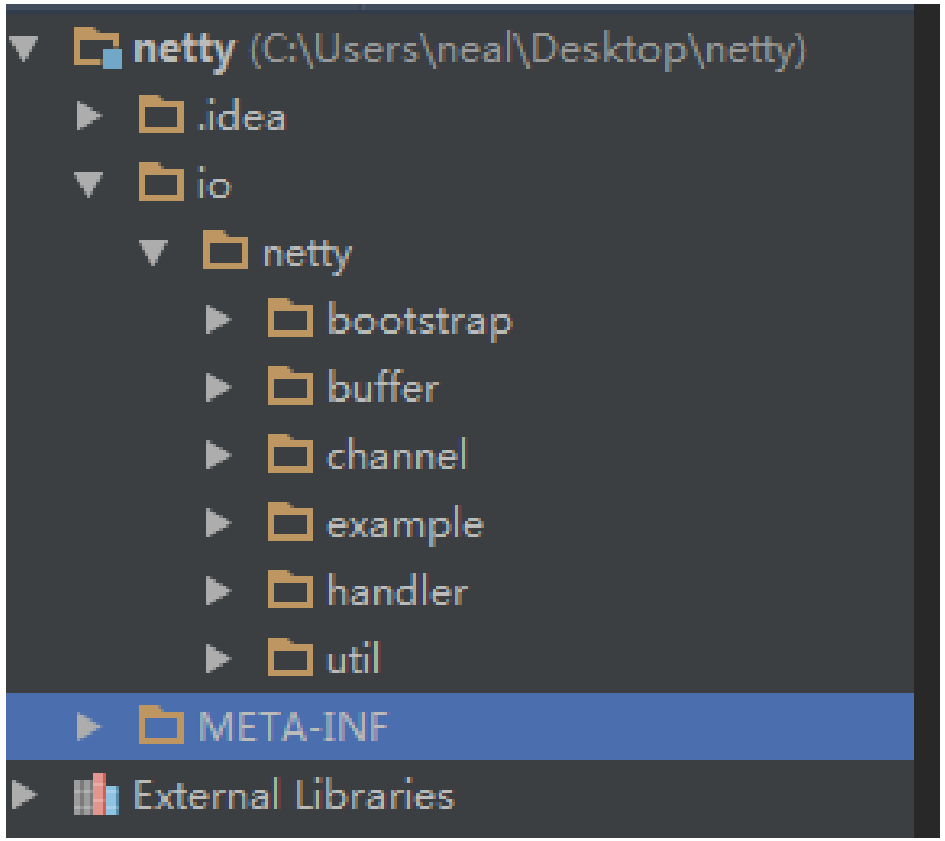


Channels
Buffers
Selectors





What does Netty look like?



Boootstrap:初始化配置
Channel：提供了不同的channel
Buffer：提供了buffer支持
Handler：io 业务处理，用户最关注的部分

How to use Netty?

```
//设置线程数
EventLoopGroup bossGroup = new NioEventLoopGroup(1);
EventLoopGroup workerGroup = new NioEventLoopGroup();
try {
    ServerBootstrap b = new ServerBootstrap();
    b.group(bossGroup, workerGroup)
        .channel(NioServerSocketChannel.class)
        .handler(new LoggingHandler(LogLevel.INFO))
        .childHandler(new HttpFileServerInitializer(sslCtx));

    Channel ch = b.bind(PORT).sync().channel();
    ch.closeFuture().sync();
} finally {
    bossGroup.shutdownGracefully();
    workerGroup.shutdownGracefully();
}
```

初始化

How to use Netty?

```
public class HttpFileServerInitializer extends ChannelInitializer<SocketChannel> {

    private final SslContext sslCtx;

    public HttpFileServerInitializer(SslContext sslCtx) { this.sslCtx = sslCtx; }

    @Override
    public void initChannel(SocketChannel ch) {
        ChannelPipeline pipeline = ch.pipeline();
        if (sslCtx != null) {
            pipeline.addLast(sslCtx.newHandler(ch.alloc()));
        }
        pipeline.addLast(new HttpServerCodec());
        pipeline.addLast(new HttpObjectAggregator(65536));
        pipeline.addLast(new ChunkedWriteHandler());
        pipeline.addLast(new HttpFileServerHandler());
    }
}
```

配置需要的Handler，顺序执行，依次提交到各个handler处理

How to use Netty?

```java
@Override
public void channelRead0(ChannelHandlerContext ctx, FullHttpRequest request) throws Exception {
    if (!request.getDecoderResult().isSuccess() || request.getMethod() != GET) {
        sendError(ctx, BAD_REQUEST);
        return;
    }
    callBackExecutionId = getCallBackExecutionId(request);
    if (callBackExecutionId == null) {
        sendError(ctx, BAD_REQUEST);
        return;
    }
    fileSystem = FileSystem.get(new Configuration());
    lastUpdatedFile = getCallBackFileStatus();
    if (lastUpdatedFile == null) {
        sendError(ctx, NO_CONTENT);
        return;
    }
    //写入响应头
    writeResponseHeaders(ctx, request);
    //写入响应内容
    writeResponseContent(ctx, request);
}
```

最后一个自定义handler，
处理业务逻辑

Thank You!