我是一段不羁的公告！
记得给芳芳这 3 个项目加油，添加一个 STAR 噢。
https://github.com/YunaiV/SpringBoot-Labs
https://github.com/YunaiV/onemall
https://github.com/YunaiV/ruoyi-vue-pro

- NETTY

精尽 Netty 面试题「最新更新时间：2023-07」

以下面试题，基于网络整理，和自己编辑。具体参考的文章，会在文末给出所有的链接。

如果胖友有自己的疑问，欢迎在星球提问，我们一起整理吊吊的 Netty 面试题的大保健。

而题目的难度，芳芳尽量按照从容易到困难的顺序，逐步下去。

BIO 是什么?

概念

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

，通信耗时，依赖网速。

创建一个新的线程进行链路处理。典型的**一请求一**

。后改良为用线程池的方式代替新增线程，被称

，同步，建立连接耗时。

言模式。

- NIO 相对于 BIO 来说一大进步。客户端和服务器之间通过 Channel 通信。NIO 可以在 Channel 进行读写操作。这些 Channel 都会被注册在 Selector 多路复用器上。Selector 通过一个线程不停的轮询这些 Channel 。找出已经准备就绪的 Channel 执行 IO 操作。
- NIO 通过一个线程轮询，实现千万个客户端的请求，这就是非阻塞 NIO 的特点。
  - 缓冲区 Buffer ： 它是 NIO 与 BIO 的一个重要区别。
    - BIO 是将数据直接写入或读取到流 Stream 对象中。

  - NIO 的数据操作都是在 Buffer 中进行的。Buffer 实际上是一个数组。Buffer 最常见的类型是ByteBuffer，另外还有 CharBuffer，ShortBuffer，IntBuffer，LongBuffer，FloatBuffer，DoubleBuffer。
  - 《精尽 Netty 源码分析 —— NIO 基础（三）之 Buffer》
- 通道 Channel ： 和流 Stream 不同，通道是双向的。NIO可以通过 Channel 进行数据的读、写和同时读写操作。
  - 通道分为两大类：一类是网络读写（SelectableChannel），一类是用于文件操作（FileChannel）。我们使用的是前者 SocketChannel 和 ServerSocketChannel ，都是SelectableChannel 的子类。
  - 《精尽 Netty 源码分析 —— NIO 基础（二）之 Channel》
- 多路复用器 Selector ： NIO 编程的基础。多路复用器提供选择已经就绪的任务的能力：就是 Selector 会不断地轮询注册在其上的通道（Channel），如果某个通道处于就绪状态，会被 Selector 轮询出来，然后通过 SelectionKey 可以取得就绪的Channel集合，从而进行后续的 IO 操作。
  - 服务器端只要提供一个线程负责 Selector 的轮询，就可以接入成千上万个客户端，这就是 JDK NIO 库的巨大进步。
  - 《精尽 Netty 源码分析 —— NIO 基础（四）之 Selector》

示例

- 代码参见 nio
- 《精尽 Netty 源码分析 —— NIO 基础（五）之示例》

小结

NIO 模型中通过 SocketChannel 和 ServerSocketChannel 实现套接字通道的通信。非阻塞，同步，避免为每个 TCP 连接创建一个线程。

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

来看《精尽 Netty 源码分析 —— NIO 基础（一）之简

述异步 IO，实际是不太正确的：

NIO 2)是异步 IO。具体原因，推荐阅

NI AIO, Netty)》 。

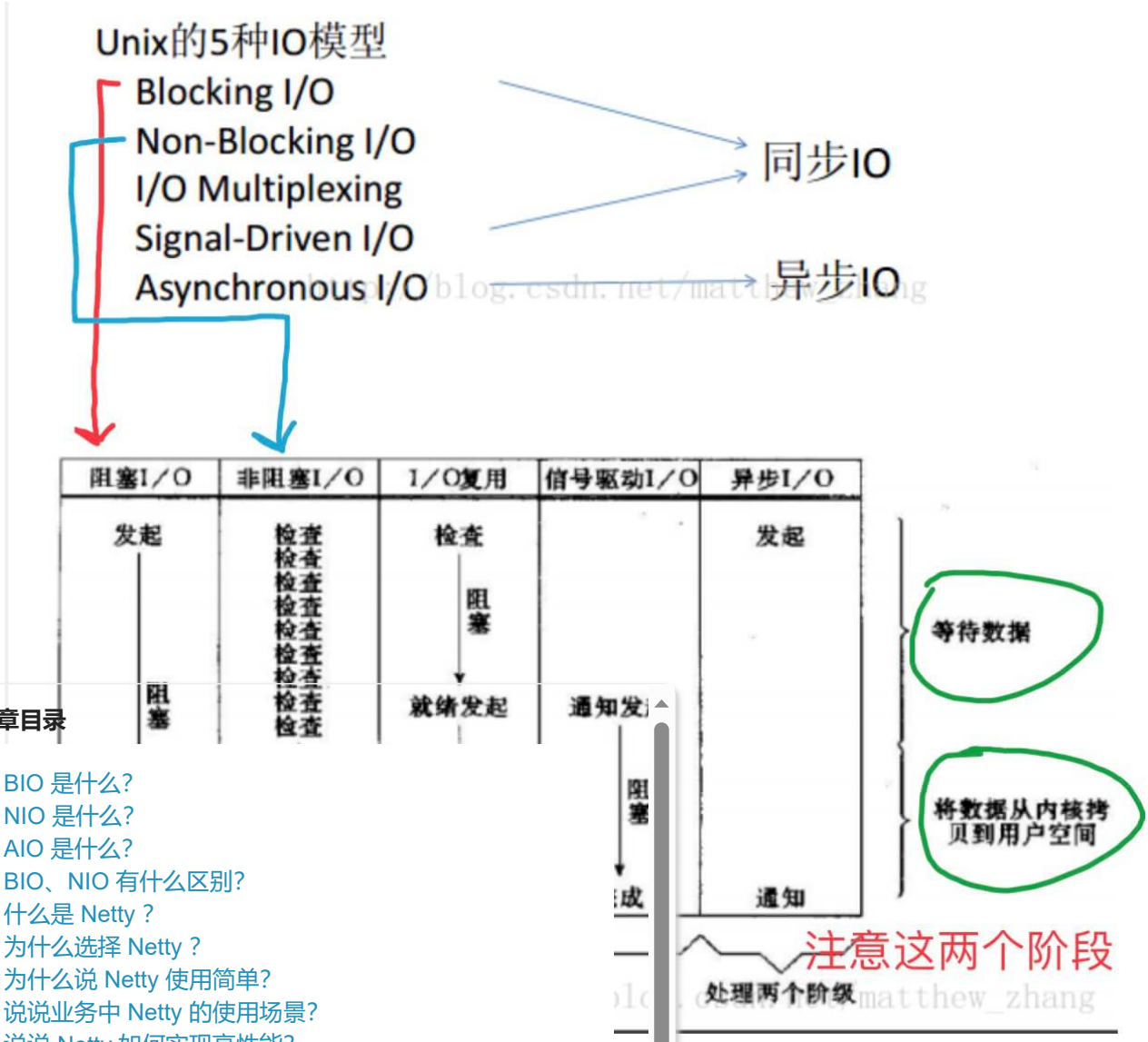
)》

要全由操作系统处理。

操后，是否有立刻返回一个标志

因此，Java NIO 是同步且非阻塞的 IO 。

- 另外，胖友在瞅瞅下面这个图来理解下：



文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

NIO 的理解，对于 AIO 来说，

信模式。在 NIO 的基础上，引入了新的异步通道

- AIO 并没有采用 NIO 的多路复用器，而是使用异步通道的概念。其 read，write 方法的返回类型，都是 Future 对象。而 Future 模型是异步的，其核心思想是：**去主函数等待时间。**

示例

- 代码参见 aio

小结

AIO 模型中通过 AsynchronousSocketChannel 和 AsynchronousServerSocketChannel 实现套接字通道的通信。非阻塞，异步。

BIO、NIO 有什么区别?

- 线程模型不同
  - BIO：一个连接一个线程，客户端有连接请求时服务器端就需要启动一个线程进行处理。所以，线程开销大。可改良为用线程池的方式代替新创建线程，被称为伪异步 IO 。
  - NIO：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新的 I/O 请求时，才启动一个线程进行处理。可改良为一个线程处理多个请求，基于 多 Reactor 模型。
- BIO 是面向流(Stream)的，而 NIO 是面向缓冲区(Buffer)的。
- BIO 的各种操作是阻塞的，而 NIO 的各种操作是非阻塞的。
- BIO 的 Socket 是单向的，而 NIO 的 Channel 是双向的。

可能文字比较难记，整理出来就是下图：

	BIO	NIO	AIO
是否阻塞	阻塞	非阻塞	非阻塞
同步/异步	同步	同步	异步
性能	低	一般	高
		: 1	N : 0
			高

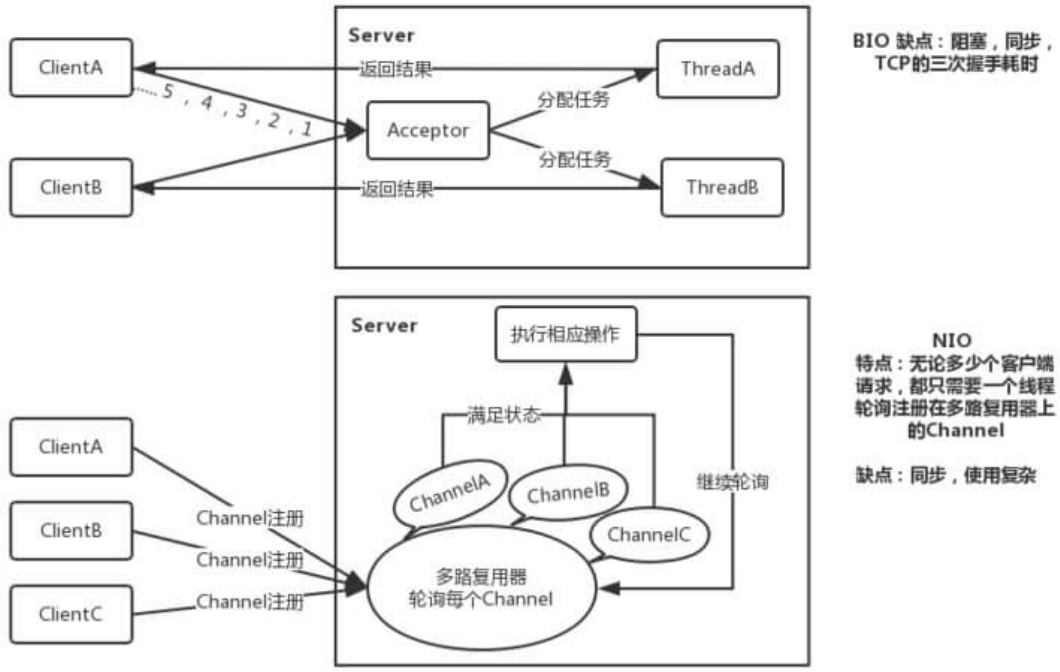
比 /

们 业务场景，NIO 和 AIO 的性能差距实际没这么
与想 中这么强悍，所以 Netty5 被废弃，而是继续保持

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

为了胖友能更好的记住和理解 BIO、NIO、AIO 的流程，胖友可以在理解下图：



文章目录

BIO NIO AIO

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

> （以上摘自百度百科）。

Netty 具有如下特性(摘自《Netty in Action》)

分类	Netty的特性
设计	1. 统一的 API，支持多种传输类型(阻塞和非阻塞的) 2. 简单而强大的线程模型 3. 真正的无连接数据报套接字(UDP)支持 4. 连接逻辑组件(ChannelHander 中顺序处理消息)以及组件复用(一个 ChannelHandel 可以被多个ChannelPipeLine 复用)
易于使用	1. 详实的 Javadoc 和大量的示例集 2. 不需要超过 JDK 1.6+ 的依赖
性能	拥有比 Java 的核心 API 更高的吞吐量以及更低的延迟(得益于池化和复用)，更低的资源消耗以及最少的内存复制
健壮性	1. 不会因为慢速、快速或者超载的连接而导致 OutOfMemoryError 2. 消除在高速网络中 NIO 应用程序常见的不公平读 / 写比率
安全性	完整的 SSL/TLS 以及 StartTLs 支持，可用于受限环境下，如 Applet 和 OSGI
社区驱动	发布快速而且频繁

为什么选择 Netty ？

文章目录

- BIO 是什么?
- NIO 是什么?
- AIO 是什么?
- BIO、NIO 有什么区别?
- 什么是 Netty ？
- 为什么选择 Netty ？
- 为什么说 Netty 使用简单?
- 说说业务中 Netty 的使用场景?
- 说说 Netty 如何实现高性能?
- Netty 的高性能如何体现?
- Netty 的高可靠如何体现?
- Netty 的可扩展如何体现?
- 简单介绍 Netty 的核心组件?
- 说说 Netty 的逻辑架构?
- 什么是 Reactor 模型?
- 请介绍 Netty 的线程模型?
- 什么是业务线程池?
- TCP 粘包 / 拆包的原因？应该这么解决?
- 了解哪几种序列化协议?
- Netty 的零拷贝实现?
- 原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
- 什么是 Netty 空闲检测?
- Netty 如何实现重连?
- Netty 自己实现的 ByteBuf 有什么优点?
- Netty 为什么要实现内存管理?

- **使用简单**：API 使用简单，开发门槛低。

扩展
能最 。
发人 不需要再为 NIO 的 BUG 而烦恼。
，更 的新功能会被加入。
在互 网、大数据、网络游戏、企业应用、电信软件等
应用

etty 也同样适用于其他技术栈。当然，😈 面试都可以酱

下：

ectc 上，监听 `SelectionKey.OP_ACCEPT` 。
。
是新 客户端接入，调用

ctor ，监听 `OP_READ` 。
包需 读取，则构造 ByteBuffer 对象，读取数据。

> 芳芳：注意噢，上述步骤还是最简的 Java NIO 启动步骤，不包括多 **Reactor** 多线程模型噢！可能有胖友不知道什么是 Reactor 模型，在 「什么是 Reactor 模型？」 问题中，我们会详细解释。

使用 **Netty 的步骤**如下：

1. 创建 NIO 线程组 EventLoopGroup 和 ServerBootstrap。
   - 设置 ServerBootstrap 的属性：线程组、SO_BACKLOG 选项，设置 NioServerSocketChannel 为 Channel
   - 设置业务处理 Handler 和 编解码器 Codec 。
   - 绑定端口，启动服务器程序。
2. 在业务处理 Handler 中，处理客户端发送的数据，并给出响应。

那么相比 Java NIO，使用 Netty 开发程序，都**简化了哪些步骤**呢？

1. 无需关心 `OP_ACCEPT` 、 `OP_READ` 、 `OP_WRITE` 等等 **IO 操作**，Netty 已经封装，对我们在使用是透明无感的。
2. 使用 boss 和 worker EventLoopGroup ， Netty 直接提供**多 Reactor 多线程模型**。
3. 在 Netty 中，我们看到有使用一个解码器 FixedLengthFrameDecoder，可以用于处理定长消息的问题，能够解决 **TCP 粘包拆包**问题，十分方便。如果使用 Java NIO ，需要我们自行实现解码器。

如果胖友不知道如何使用 Java NIO 编写一个 Server ，建议自己去实现以下。 如果胖友没有使用过 Netty 编写一个 Server ，建议去入门下。

说说业务中 Netty 的使用场景？

- 构建高性能、低时延的各种 Java 中间件，Netty 主要作为基础通信[illegible]提供高性能、低时延的通信服务。例如：
- RocketMQ ，分布式消息队列。

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

异步 高性能的 WebSocket、Protobuf 等协议的支持。

用 内部各模块的数据分发、传输和汇总等，实现模块

用 entLoop 等创新性的机制，可以非常高效地管理

的 ，减少了对象的创建和销毁。并且，内吃吃的内

据 uffer 从一个内存区域拷贝到另

，此 CPU 的效率就得到的提

4. **协议支持** ： 提供对 Protobuf 等高性能序列化协议支持。

5. **使用更多本地代码**。例如：

   - 直接利用 JNI 调用 Open SSL 等方式，获得比 Java 内建 SSL 引擎更好的性能。
   - 利用 JNI 提供了 Native Socket Transport ， 在使用 Epoll edge-triggered 的情况下，可以有一定的性能提升。

6. 其它：

   - 利用反射等技术直接操纵 SelectionKey ， 使用数组而不是 Java 容器等。

- 实现 FastThreadLocal 类，当请求频繁时，带来更好的性能。
- ....

另外，推荐阅读白衣大大的两篇文章：

1. 《Netty高性能编程备忘录(上)》
2. 《Netty高性能编程备忘录（下）》

> 下面三连问！
>
> Netty 是一个高性能的、高可靠的、可扩展的异步通信框架，那么高性能、高可靠、可扩展设计体现在哪里呢？

Netty 的高性能如何体现？

> 这个问题，和「说说 Netty 如何实现高性能？」问题，会有点重叠。没事，反

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

如何实现高性能的呢?

了传统同步阻塞 I/O 模式下服务端无法平滑处理

内存复制，提升了 I/O 读取和写入性能。
建和销毁 ByteBuf 带来的性能消耗。
制化的调优参数，满足不同的性能场景。
程安全容器或锁。

发访问带来的锁竞争和额外的 CPU 资源消耗问

的内存管理降低了 GC 的频率，减少频繁 GC 带

要在消息完成交互时关闭链路，因此相对于短连
期性地进行链路检测。使用心跳机制的原因是，
致消息积压无法处理。为了解决这个问题，需要
重建 TCP 连接。为了支持心跳，Netty 提供了两种链路空闲检测机制：

- 读空闲超时机制：连续 T 周期没有消息可读时，发送心跳消息，进行链路检测。如果连续 N 个周期没有读取到心跳消息，可以主动关闭链路，重建连接。
- 写空闲超时机制：连续 T 周期没有消息需要发送时，发送心跳消息，进行链路检测。如果连续 N 个周期没有读取对方发回的心跳消息，可以主动关闭链路，重建连接。
- 《精尽 Netty 源码解析 —— ChannelHandler（五）之 IdleStateHandler》

2. **内存保护机制**：Netty 提供多种机制对内存进行保护，包括以下几个方面：
   - 通过对象引用计数器对 ByteBuf 进行细粒度的内存申请和释放，对非法的对象引用进行检测和保护

- 可设置的内存容量上限，包括 ByteBuf、线程池线程数等，避免异常请求耗光内存。

3. **优雅停机**：优雅停机功能指的是当系统推出时，JVM 通过注册的 Shutdown Hook 拦截到退出信号量，然后执行推出操作，释放相关模块的资源占用，将缓冲区的消息处理完成或清空，将待刷新的数据持久化到磁盘和数据库中，等到资源回收和缓冲区消息处理完成之后，再退出。
   - 《精尽 Netty 源码解析 —— EventLoop（八）之 EventLoop 优雅关闭》

Netty 的可扩展如何体现?

可定制、易扩展。

- **责任链模式**：ChannelPipeline 基于责任链模式开发，便于业务逻辑的拦截、定制和扩展。
- **基于接口的开发**：关键的类库都提供了接口或抽象类，便于用户自定义实现。
- **提供大量的工厂类**：通过重载这些工厂类，可以按需创建出用户需要的对象。
- **提供大量系统参数**：供用户按需设置，增强系统的场景定制性。

> 芳芳：说个题外话。
>
> 实际上，任何的技术的研究，我们都可以去思考，它的高性能是怎么体现的，它的可靠性是怎么体现的，它的可扩展是怎么体现的。

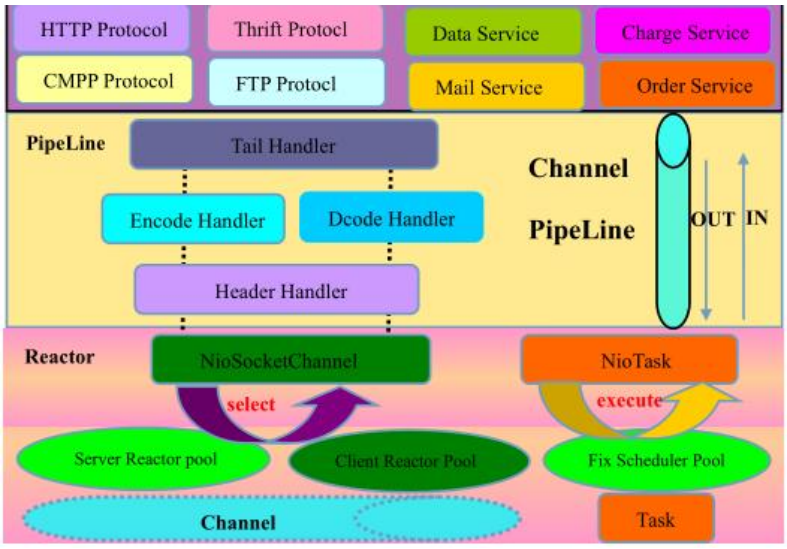
文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

详细的，请直接阅读《精尽 Netty 源码分析 —— Netty 简介（二）之核心组件》一文。

说说 Netty 的逻辑架构?

Netty 采用了典型的**三层网络架构**进行设计和开发，其逻辑架构如下图所示：



Netty 逻辑架构图

> 芳芳：注意，这个图是自下向上看。哈哈哈~

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

...entLoop 及其父类，NioSocketChannel 和...接操作，负责将网络层的数据读到内存缓冲区，然...将这些事件触发到 pipeline 中，由 pipeline 管理...

...动态地编排职责链。职责链可以选择监听和处理...

...编排，一类是应用层协议插件，用于特定协议相...且变动较小，故而将应用协议到 POJO 的转变和...辑层的隔离，实现架构层面的分层隔离。

...模型...一文。

...模型...一文。

...都需要在 Reactor 所在的线程中，进行读写操作。那么此时就会有一个问题，如果我们读取到数据，需要进行业务逻辑处理，并且这个业务逻辑需要对数据库、缓存等等进行操作，会有什么问题呢？假设这个数据库操作需要 5 ms ，那就意味着这个 Reactor 线程在这 5 ms 无法进行注册在这个 Reactor 的 Channel 进行读写操作。也就是说，多个 Channel 的所有读写操作都变成了串行。势必，这样的效率会非常非常非常的低。

解决

那么怎么解决呢？创建业务线程池，将读取到的数据，提交到业务线程池中进行处理。这样，Reactor 的 Channel 就不会被阻塞，而 Channel 的所有读写操作都变成了并行了。

案例

如果胖友熟悉 Dubbo 框架，就会发现 《Dubbo 用户指南 —— 线程模型》 。😈 认真读下，可以跟面试官吹一吹啦。

TCP 粘包 / 拆包的原因？应该这么解决？

概念

TCP 是以流的方式来处理数据，所以会导致粘包 / 拆包。

- 拆包：一个完整的包可能会被 TCP 拆分成多个包进行发送。
- 粘包：也可能把小的封装成一个大的数据包发送。

原因

- 应用程序写入的字节大小大于套接字发送缓冲区的大小，会发生**拆包**现象。而应用程序写入数据小于套接字缓冲区大小，网卡将应用多次写入的数据发送到网络上，这将会发生**粘包**现象。
- 待发送数据大于 MSS（最大报文长度），TCP 在传输前将进行**拆包**。
- 以太网帧的 payload（净荷）大于 MTU（默认为 1500 字节）进行 IP 分片**拆包**。
- 接收数据端的应用层没有及时读取接收缓冲区中的数据，将发生**粘包**

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

包处 的。
进行 包拆包处理的。
理的
粘包 包处理的。

(一 Cumulator》
(二 FrameDecoder》

主要 于网络传输、数据持久化等。
成原 对象，主要用于网络传输对象的解码，以便完成远程调用。

选型

在选择序列化协议的选择，主要考虑以下三个因素：

- 序列化后的**字节大小**。更少的字节数，可以减少网络带宽、磁盘的占用。
- 序列化的**性能**。对 CPU、内存资源占用情况。

- 是否支持**跨语言**。例如，异构系统的对接和开发语言切换。

方案

> 如果对序列化工具了解不多的胖友，可能一看有这么多优缺点会比较懵逼，可以先记得有哪些序列化工具，然后在慢慢熟悉它们的优缺点。
>
> 重点，还是知道【选型】的考虑点。

1. 【重点】Java 默认提供的序列化
   - 无法跨语言；序列化后的字节大小太大；序列化的性能差。
2. 【重点】XML 。
   - 优点：人机可读性好，可指定元素或特性的名称。
   - 缺点：序列化数据只包含数据本身以及类的结构，不包括类型标识和程序集信息；只能序列化公共属性和字段；不能序列化方法；文件庞大，文件格式复杂，传输占带宽。
   - 适用场景：当做配置文件存储数据，实时数据转换。
3. 【重点】JSON ，是一种轻量级的数据交换格式。
   - 优点：兼容性高、数据格式比较简单，易于读写、序列化后数据
   ，可扩展性好，兼容性好。与 XML 相比，其

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因? 应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么? Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

的情 额外空间开销比较大。
、基 Restful API 请求、传输数据量相对小，实时性

据类 、对于数据字段的增删具有较强的兼容性、支

，i 代码时相对困难、不能与其他传输层协议共同
适 数据持久化序列化协议。

没有 的问题。
自 述属性、提高了数据解析速度、快速可压缩的
语言 现。

化 格式。
过代 生成工具可以生成对应数据结构的 POJO 对象和

JS 等）、通过标识字段的顺序，可以实现协议的

只 Java 、C++、python。
的议 属性、适合应用层对象的持久化。

文 直接导包即可。

读 va.io.Serializable 接口。

- 【了解】Message Pack ，一个高效的二进制序列化格式。
- 【重点】Hessian ，采用二进制协议的轻量级 remoting on http 服务。
  - 目前，阿里 RPC 框架 Dubbo 的**默认**序列化协议。
- 【重要】kryo ，是一个快速高效的Java对象图形序列化框架，主要特点是性能、高效和易用。该项目用来序列化对象到文件、数据库或者网络。
  - 目前，阿里 RPC 框架 Dubbo 的可选序列化协议。

- 【重要】FST ，fast-serialization 是重新实现的 Java 快速对象序列化的开发包。序列化速度更快（2-10倍）、体积更小，而且兼容 JDK 原生的序列化。要求 JDK 1.7 支持。
  - 目前，阿里 RPC 框架 Dubbo 的可选序列化协议。

Netty 的零拷贝实现?

Netty 的零拷贝实现，是体现在多方面的，主要如下：

1. 【重点】Netty 的接收和发送 ByteBuffer 采用堆外直接内存 **Direct Buffer** 。
   - 使用堆外直接内存进行 Socket 读写，不需要进行字节缓冲区的二次拷贝；使用堆内内存会多了一次内存拷贝，JVM 会将堆内存 Buffer 拷贝一份到直接内存中，然后才写入 Socket 中。
   - Netty 创建的 ByteBuffer 类型，由 ChannelConfig 配置。而 ChannelConfig 配置的 ByteBufAllocator 默认创建 Direct Buffer 类型。
2. **CompositeByteBuf** 类，可以将多个 ByteBuf 合并为一个逻辑上的 ByteBuf ，避免了传统通过内存拷贝的方式将几个小 Buffer 合并成一个大的 Buffer 。
   - `#addComponents(...)` 方法，可将 header 与 body 合并为一个逻辑上的 ByteBuf 。这两个 ByteBuf 在 CompositeByteBuf 内部都是单独存在的，即 CompositeByteBuf 只是逻辑上是一个整体。
3. 通过 **FileRegion** 包装的 FileChannel 。
   - `#tranferTo(...)` 方法，实现文件传输, 可以直接将文件缓冲区的数据发送到目标 Channel ，避免了传统通过循环 write 方式，导致的内存拷贝问题。
4. 通过 **wrap** 方法, 我们可以将 `byte[]` 数组、ByteBuf、ByteBuffer 等包装成一个 Netty ByteBuf 对象, 进而避免了拷贝

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

IdleStateHandler 目前提供三种类型的心跳检测，通过构造方法来设置。代码如下：

```
// IdleStateHandler.java

public IdleStateHandler(
        int readerIdleTimeSeconds,
```

```
        int writerIdleTimeSeconds,
        int allIdleTimeSeconds) {
    this(readerIdleTimeSeconds, writerIdleTimeSeconds, allIdleTimeSeconds,
         TimeUnit.SECONDS);
}
```

- `readerIdleTimeSeconds` 参数：为读超时时间，即测试端一定时间内未接受到被测试端消息。
- `writerIdleTimeSeconds` 参数：为写超时时间，即测试端一定时间内向被测试端发送消息。
- `allIdleTimeSeconds` 参数：为读或写超时时间。

另外，我们会在网络上看到类似《IdleStateHandler 心跳机制》这样标题的文章，实际上空闲检测和心跳机制是**两件事**。

- 只是说，因为我们使用 IdleStateHandler 的目的，就是检测到连接处于空闲，通过心跳判断其是否还是**有效的连接**。
- 虽然说，TCP 协议层提供了 Keeplive 机制，但是该机制默认的心跳时间是 2 小时，依赖操作系统实现不够灵活。因而，我们才在应用层上，自己实现心跳机制。

具体的，我们来看看下面的问题 「Netty 如何实现重连？」 。

Netty 如何实现重连？

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

15

接。

例如 90 秒。

客户端。通过这样的方式，使客户端知道自己心跳成

。核心代码如下：

```
.TRANSPORT_CLIENT_IDLE, TaroConstants.TRAN
```

er。核心代码如下：

```
stants.TRANSPORT_SERVER_IDLE, TimeUnit.MIL
```

- ClientHeartbeatHandler.java 中，碰到空闲，则发起心跳。不过，如何重连，暂时没有实现。需要考虑，重新发起连接可能会失败的情况。具体的，可以看看 《一起学Netty（十四）之 Netty生产级的心跳和重连机制》 文章中的，ConnectionWatchdog 的代码。
- ServerHeartbeatHandler.java 中，检测到客户端空闲，则直接关闭连接。

Netty 自己实现的 ByteBuf 有什么优点?

如下是 《Netty 实战》 对它的**优点总**结:

> - A01. 它可以被用户自定义的缓冲区类型扩展
> - A02. 通过内置的符合缓冲区类型实现了透明的零拷贝
> - A03. 容量可以按需增长
> - A04. 在读和写这两种模式之间切换不需要调用 `#flip()` 方法
> - A05. 读和写使用了不同的索引
> - A06. 支持方法的链式调用
> - A07. 支持引用计数
> - A08. 支持池化

特别是第 A04 这点，相信很多胖友都被 NIO ByteBuffer 反人类的读式和写模式给坑哭了。在 《精尽 Netty 源码分析 —— NIO 基础（三）之 Buffer》 中，我们也吐槽过了。

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

z By Buf（一）简介》 。

络传 性能，Direct ByteBuffer 必然是最合适的选择。但是
里， 次重用是比较有效的方式。但是，不同于一般于我们
又但 ，申请多大的 Direct ByteBuffer 进行池化又会是一个
！ 以呢，就需要有一个合适的内存管理算法，解决**高效**

r》

性的 buffer 池，分配策略则是结合
em **lloc** 变种，代码在

> - 频繁分配、释放 buffer 时减少了 GC 压力。
> - 在初始化新 buffer 时减少内存带宽消耗(初始化时不可避免的要给buffer数组赋初始值)。

- 及时的释放 direct buffer 。

hushi55 大佬的理解

> C/C++ 和 java 中有个围城，城里的想出来，城外的想进去！ **
>
> 这个围城就是自动内存管理！

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

Netty 4 buffer 介绍

Bu 的实现，相比之下，通过维护两
fe ByteBuf 简单不少，也会更高
的 大不同，就是他不再基于传统
C+ 中的 malloc/free 的机制，需要
内存 管理上升到GC，是一个历史的巨
回归 到了手动内存管理模式，正印证
的，没有永远的最好。

值是 不言而喻的，不仅大大的降低了
了内 存管理带来的 Crash 困扰，为函
编程 带来了春天。 并且，高效的GC算
法效 率。 不过，也有很多的情况，可

路由转发型应用（很多erlang应用
比如 1M级，在这种情况下，在每次
是基本都使产生1K的垃圾，都会导致频繁的GC产生。 在这种模式下，
erlang 的按进程回收模式，或者是 C/C++ 的手工回收机制，效率更高。
- Cache 型应用，由于对象的存在周期太长，GC 基本上就变得没有价值。

所以，理论上，尴尬的GC实际上比较适合于处理介于这 2 者之间的情况： 对象分配的频繁程度相比数据处理的时间要少得多的，但又是相对短暂的， 典型的，对于OLTP型的服务，处理能力在 1K QPS 量级，每个请求的对象分配在 10K-50K 量级， 能够在 5-10s 的时间内进行一 次younger GC ，每次GC的时间可以控制在 10ms 水平上， 这类的应用，实在是太适合 GC 行的模式了，而且结合 Java 高效的分代 GC ，简直就是一个理想搭配。

② 影响

Netty 4 引入了手工内存的模式，我觉得这是一大创新，这种模式甚至于会延展， 应用到 Cache 应用中。实际上，结合 JVM 的诸多优秀特性，如果用 Java 来实现一个 Redis 型 Cache、 或者 In-memory SQL Engine，或者是一个 Mongo DB，我觉得相比 C/C++ 而言，都要更简单很多。 实际上，JVM 也已经提供了打通这种技术的机制，就是 Direct Memory 和 Unsafe 对象。 基于这个基础，我们可以像 C 语言一样直接操作内存。实际上，Netty4 的 ByteBuf 也是基于这个基础的。

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?

之 malloc（一）简介》。

去看《精尽 Netty 源码解析 —— 比较难和复杂。

而每 Chunk 默认由2048个page组成。
Pag，而中间节点表示内存区域，节点自己记录它在
标记会被标记，这样就表示这个中间节点以下的所有
而 **PoolSubpage** 用于分配小于 8k 的内存，它会把一

现的?

芳芳：这是一道比较复杂的面试题，可以挑战一下。

推荐阅读如下两篇文章：

- 《Netty 之有效规避内存泄漏》从原理层面解释。

- 《精尽 Netty 源码解析 —— Buffer 之 ByteBuf（三）内存泄露检测》 从源码层面解读。

另外，Netty 的内存泄露检测的实现，是对 WeakReference 和 ReferenceQueue 很好的学习。之前很多胖友在看了 《Java 中的四种引用类型》 之后，是不太理解 Java 的四种引用的具体使用场景，这不就来了么。

666. 彩蛋

😈 撸完 Netty 源码之后，一段时间没去用，东西都忘了蛮多。不过因为读过源码，在看这些面试题，会发现轻松太多了。有一种，“老朋友”的感觉。妥妥的。

参考与推荐如下文章：

- itdragon 《Netty 序章之 BIO NIO AIO 演变》
- 白夜行515 《【面试题】Netty相关》
- albon 《Netty 权威指南笔记（二）：Java NIO 和 Netty 对比》
- albon 《Netty 权威指南笔记（四）：架构剖析》
- 狗窝 《面试题之 Netty》 题目很不错，就是图片都挂了。

文章目录

BIO 是什么?
NIO 是什么?
AIO 是什么?
BIO、NIO 有什么区别?
什么是 Netty ?
为什么选择 Netty ?
为什么说 Netty 使用简单?
说说业务中 Netty 的使用场景?
说说 Netty 如何实现高性能?
Netty 的高性能如何体现?
Netty 的高可靠如何体现?
Netty 的可扩展如何体现?
简单介绍 Netty 的核心组件?
说说 Netty 的逻辑架构?
什么是 Reactor 模型?
请介绍 Netty 的线程模型?
什么是业务线程池?
TCP 粘包 / 拆包的原因？应该这么解决?
了解哪几种序列化协议?
Netty 的零拷贝实现?
原生的 NIO 存在 Epoll Bug 是什么？Netty 是怎么解决的?
什么是 Netty 空闲检测?
Netty 如何实现重连?
Netty 自己实现的 ByteBuf 有什么优点?
Netty 为什么要实现内存管理?