04 | 网络通信：RPC框架在网络通信上更倾向于哪种网络IO模型？

2020-02-26 何小锋

RPC实战与核心原理 进入课程 >

讲述：张浩
时长 13:29 大小 15.45M

你好，我是何小锋。在上一讲我讲解了 RPC 框架中的序列化，通过上一讲，我们知道由于网络传输的数据都是二进制数据，所以我们要传递对象，就必须将对象进行序列化，而 RPC 框架在序列化的选择上，我们更关注序列化协议的安全性、通用性、兼容性，其次才关注序列化协议的性能、效率、空间开销。承接上一讲，这一讲，我要专门讲解下 RPC 框架中的网络通信，这也是我们在开篇词中就强调过的重要内容。

那么网络通信在 RPC 调用中起到什么作用呢？

我在[第 01 讲]中讲过，RPC 是解决进程间通信的一种方式。一次 RPC 调用，本质就是服务消费者与服务提供者间的一次网络信息交换的过程。服务调用者通过网络 IO 发送一条请求消息，服务提供者接收并解析，处理完相关的业务逻辑之后，再发送一条响应消息给服

务调用者，服务调用者接收并解析响应消息，处理完相关的响应逻辑，一次 RPC 调用便结束了。可以说，网络通信是整个 RPC 调用流程的基础。

常见的网络 IO 模型

那说到网络通信，就不得不提一下网络 IO 模型。为什么要讲网络 IO 模型呢？因为所谓的两台 PC 机之间的网络通信，实际上就是两台 PC 机对网络 IO 的操作。

常见的网络 IO 模型分为四种：同步阻塞 IO（BIO）、同步非阻塞 IO（NIO）、IO 多路复用和异步非阻塞 IO（AIO）。在这四种 IO 模型中，只有 AIO 为异步 IO，其他都是同步 IO。

其中，最常用的就是同步阻塞 IO 和 IO 多路复用，这一点通过了解它们的机制，你会 get 到。至于其他两种 IO 模型，因为不常用，则不作为本讲的重点，有兴趣的话我们可以在留言区中讨论。

阻塞 IO（blocking IO）

同步阻塞 IO 是最简单、最常见的 IO 模型，在 Linux 中，默认情况下所有的 socket 都是 blocking 的，先看下操作流程。

首先，应用进程发起 IO 系统调用后，应用进程被阻塞，转到内核空间处理。之后，内核开始等待数据，等待到数据之后，再将内核中的数据拷贝到用户内存中，整个 IO 处理完毕后返回进程。最后应用的进程解除阻塞状态，运行业务逻辑。

这里我们可以看到，系统内核处理 IO 操作分为两个阶段——等待数据和拷贝数据。而在这两个阶段中，应用进程中 IO 操作的线程会一直都处于阻塞状态，如果是基于 Java 多线程开发，那么每一个 IO 操作都要占用线程，直至 IO 操作结束。

这个流程就好比我们去餐厅吃饭，我们到达餐厅，向服务员点餐，之后要一直在餐厅等待后厨将菜做好，然后服务员会将菜端给我们，我们才能享用。

IO 多路复用（IO multiplexing）

多路复用 IO 是在高并发场景中使用最为广泛的一种 IO 模型，如 Java 的 NIO、Redis、Nginx 的底层实现就是此类 IO 模型的应用，经典的 Reactor 模式也是基于此类 IO 模型。

那么什么是 IO 多路复用呢？通过字面上的理解，多路就是指多个通道，也就是多个网络连接的 IO，而复用就是指多个通道复用在一个复用器上。

多个网络连接的 IO 可以注册到一个复用器（select）上，当用户进程调用了 select，那么整个进程会被阻塞。同时，内核会“监视”所有 select 负责的 socket，当任何一个 socket 中的数据准备好了，select 就会返回。这个时候用户进程再调用 read 操作，将数据从内核中拷贝到用户进程。

这里我们可以看到，当用户进程发起了 select 调用，进程会被阻塞，当发现该 select 负责的 socket 有准备好的数据时才返回，之后才发起一次 read，整个流程要比阻塞 IO 要复杂，似乎也更浪费性能。但它最大的优势在于，用户可以在一个线程内同时处理多个 socket 的 IO 请求。用户可以注册多个 socket，然后不断地调用 select 读取被激活的 socket，即可达到在同一个线程内同时处理多个 IO 请求的目的。而在同步阻塞模型中，必须通过多线程的方式才能达到这个目的。

同样好比我们去餐厅吃饭，这次我们是几个人一起去的，我们专门留了一个人在餐厅排号等位，其他人就去逛街了，等排号的朋友通知我们可以吃饭了，我们就直接去享用了。

为什么说阻塞 IO 和 IO 多路复用最为常用?

了解完二者的机制，我们就可以回到起初的问题了——我为什么说阻塞 IO 和 IO 多路复用最为常用。对比这四种网络 IO 模型：阻塞 IO、非阻塞 IO、IO 多路复用、异步 IO。实际在网络 IO 的应用上，需要的是系统内核的支持以及编程语言的支持。

在系统内核的支持上，现在大多数系统内核都会支持阻塞 IO、非阻塞 IO 和 IO 多路复用，但像信号驱动 IO、异步 IO，只有高版本的 Linux 系统内核才会支持。

在编程语言上，无论 C++ 还是 Java，在高性能的网络编程框架的编写上，大多数都是基于 Reactor 模式，其中最为典型的便是 Java 的 Netty 框架，而 Reactor 模式是基于 IO 多路复用的。当然，在非高并发场景下，同步阻塞 IO 是最为常见的。

综合来讲，在这四种常用的 IO 模型中，应用最多的、系统内核与编程语言支持最为完善的，便是阻塞 IO 和 IO 多路复用。这两种 IO 模型，已经可以满足绝大多数网络 IO 的应用场景。

RPC 框架在网络通信上倾向选择哪种网络 IO 模型?

讲完了这两种最常用的网络 IO 模型，我们可以看看它们都适合什么样的场景。

IO 多路复用更适合高并发的场景，可以用较少的进程（线程）处理较多的 socket 的 IO 请求，但使用难度比较高。当然高级的编程语言支持得还是比较好的，比如 Java 语言有很多的开源框架对 Java 原生 API 做了封装，如 Netty 框架，使用非常简便；而 GO 语言，语言本身对 IO 多路复用的封装就已经很简洁了。

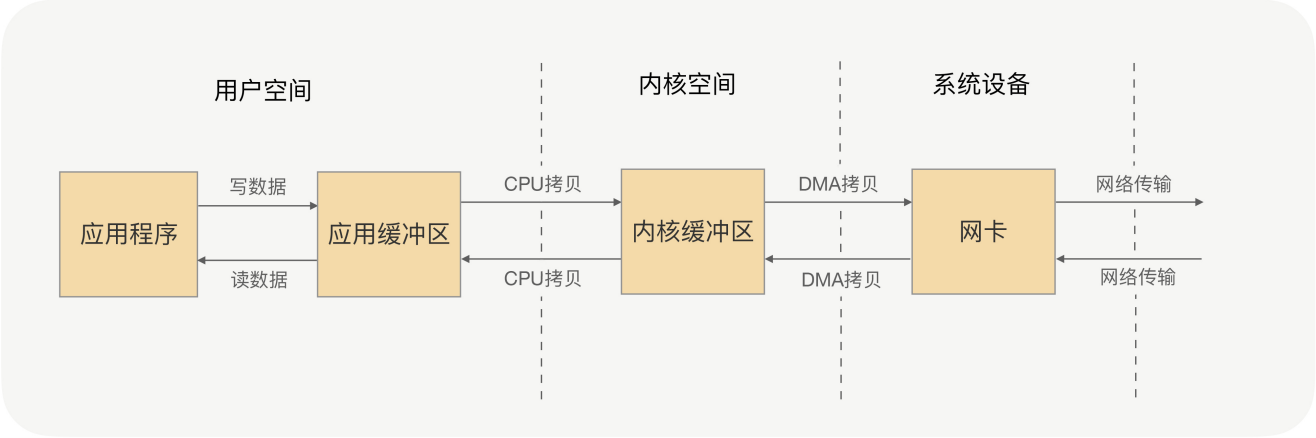
而阻塞 IO 与 IO 多路复用相比，阻塞 IO 每处理一个 socket 的 IO 请求都会阻塞进程（线程），但使用难度较低。在并发量较低、业务逻辑只需要同步进行 IO 操作的场景下，阻塞 IO 已经满足了需求，并且不需要发起 select 调用，开销上还要比 IO 多路复用低。

RPC 调用在大多数的情况下，是一个高并发调用的场景，考虑到系统内核的支持、编程语言的支持以及 IO 模型本身的特点，在 RPC 框架的实现中，在网络通信的处理上，我们会选择 IO 多路复用的方式。开发语言的网络通信框架的选型上，我们最优的选择是基于 Reactor 模式实现的框架，如 Java 语言，首选的框架便是 Netty 框架（Java 还有很多其他 NIO 框架，但目前 Netty 应用得最为广泛），并且在 Linux 环境下，也要开启 epoll 来提升系统性能（Windows 环境下是无法开启 epoll 的，因为系统内核不支持）。

了解完以上内容，我们可以继续看这样一个关键问题——零拷贝。在我们应用的过程中，他是非常重要的。

什么是零拷贝?

刚才讲阻塞 IO 的时候我讲到，系统内核处理 IO 操作分为两个阶段——等待数据和拷贝数据。等待数据，就是系统内核在等待网卡接收到数据后，把数据写到内核中；而拷贝数据，就是系统内核在获取到数据后，将数据拷贝到用户进程的空间中。以下是具体流程：



网络IO读写流程

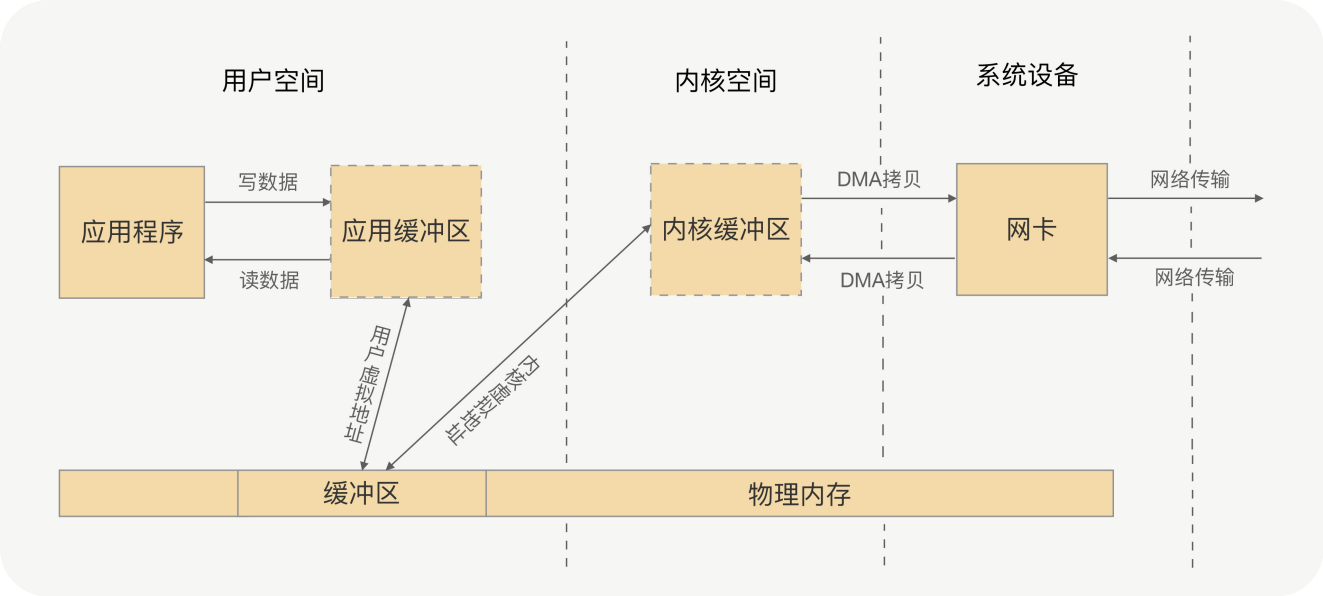
应用进程的每一次写操作，都会把数据写到用户空间的缓冲区中，再由 CPU 将数据拷贝到系统内核的缓冲区中，之后再由 DMA 将这份数据拷贝到网卡中，最后由网卡发送出去。这里我们可以看到，一次写操作数据要拷贝两次才能通过网卡发送出去，而用户进程的读操作则是将整个流程反过来，数据同样会拷贝两次才能让应用程序读取到数据。

应用进程的一次完整的读写操作，都需要在用户空间与内核空间中来回拷贝，并且每一次拷贝，都需要 CPU 进行一次上下文切换（由用户进程切换到系统内核，或由系统内核切换到用户进程），这样是不是很浪费 CPU 和性能呢？那有没有什么方式，可以减少进程间的数据拷贝，提高数据传输的效率呢？

这时我们就需要零拷贝（Zero-copy）技术。

所谓的零拷贝，就是取消用户空间与内核空间之间的数据拷贝操作，应用进程每一次的读写操作，可以通过一种方式，直接将数据写入内核或从内核中读取数据，再通过 DMA 将内核中的数据拷贝到网卡，或将网卡中的数据 copy 到内核。

那怎么做到零拷贝？你想一下是不是用户空间与内核空间都将数据写到一个地方，就不需要拷贝了？此时你有没有想到虚拟内存？



虚拟内存

零拷贝有两种解决方式，分别是 mmap+write 方式和 sendfile 方式，其核心原理都是通过虚拟内存来解决的。这两种实现方式都不难，市面上可查阅的资料也很多，在此就不详述了，有问题，可以在留言区中解决。

Netty 中的零拷贝

了解完零拷贝，我们再看看 Netty 中的零拷贝。

我刚才讲到，RPC 框架在网络通信框架的选型上，我们最优的选择是基于 Reactor 模式实现的框架，如 Java 语言，首选的便是 Netty 框架。那么 Netty 框架是否也有零拷贝机制呢？Netty 框架中的零拷贝和我之前讲的零拷贝又有什么不同呢？

刚才我讲的零拷贝是操作系统层面上的零拷贝，主要目标是避免用户空间与内核空间之间的数据拷贝操作，可以提升 CPU 的利用率。

而 Netty 的零拷贝则不大一样，他完全站在了用户空间上，也就是 JVM 上，它的零拷贝主要是偏向于数据操作的优化上。

那么 Netty 这么做的意义是什么呢?

回想下[第 02 讲]，在这一讲中我讲解了 RPC 框架如何去设计协议，其中我讲到：在传输过程中，RPC 并不会把请求参数的所有二进制数据整体一下子发送到对端机器上，中间

可能会拆分成好几个数据包，也可能会合并其他请求的数据包，所以消息都需要有边界。那么一端的机器收到消息之后，就需要对数据包进行处理，根据边界对数据包进行分割和合并，最终获得一条完整的消息。

那收到消息后，对数据包的分割和合并，是在用户空间完成，还是在内核空间完成的呢？

当然是在用户空间，因为对数据包的处理工作都是由应用程序来处理的，那么这里有没有可能存在数据的拷贝操作？可能会存在，当然不是在用户空间与内核空间之间的拷贝，是用户空间内部内存中的拷贝处理操作。Netty 的零拷贝就是为了解决这个问题，在用户空间对数据操作进行优化。

那么 Netty 是怎么对数据操作进行优化的呢？

- Netty 提供了 CompositeByteBuf 类，它可以将多个 ByteBuf 合并为一个逻辑上的 ByteBuf，避免了各个 ByteBuf 之间的拷贝。
- ByteBuf 支持 slice 操作，因此可以将 ByteBuf 分解为多个共享同一个存储区域的 ByteBuf，避免了内存的拷贝。
- 通过 wrap 操作，我们可以将 byte[] 数组、ByteBuf、ByteBuffer 等包装成一个 Netty ByteBuf 对象, 进而避免拷贝操作。

Netty 框架中很多内部的 ChannelHandler 实现类，都是通过 CompositeByteBuf、slice、wrap 操作来处理 TCP 传输中的拆包与粘包问题的。

那么 Netty 有没有解决用户空间与内核空间之间的数据拷贝问题的方法呢？

Netty 的 ByteBuffer 可以采用 Direct Buffers，使用堆外直接内存进行 Socketd 的读写操作，最终的效果与我刚才讲解的虚拟内存所实现的效果是一样的。

Netty 还提供 FileRegion 中包装 NIO 的 FileChannel.transferTo() 方法实现了零拷贝，这与 Linux 中的 sendfile 方式在原理上也是一样的。

总结

今天我们详细地介绍了阻塞 IO 与 IO 多路复用，拓展了零拷贝相关的知识以及 Netty 框架中的零拷贝。

考虑到系统内核的支持、编程语言的支持以及 IO 模型本身的特点，RPC 框架在网络通信的处理上，我们更倾向选择 IO 多路复用的方式。

零拷贝带来的好处就是避免没必要的 CPU 拷贝，让 CPU 解脱出来去做其他的事，同时也减少了 CPU 在用户空间与内核空间之间的上下文切换，从而提升了网络通信效率与应用程序的整体性能。

而 Netty 的零拷贝与操作系统的零拷贝是有些区别的，Netty 的零拷贝偏向于用户空间中对数据操作的优化，这对处理 TCP 传输中的拆包粘包问题有着重要的意义，对应用程序处理请求数据与返回数据也有重要的意义。

在 RPC 框架的开发与使用过程中，我们要深入了解网络通信相关的原理知识，尽量做到零拷贝，如使用 Netty 框架；我们要合理使用 ByteBuf 子类，做到完全零拷贝，提升 RPC 框架的整体性能。

课后思考

回想一下，你所接触的开源中间件框架有哪些框架在网络通信上做到了零拷贝？都是使用哪种方式实现的零拷贝？

欢迎留言和我分享你的思考和疑惑，也欢迎你把文章分享给你的朋友，邀请他加入学习。我们下节课再见！

上一篇　03 | 序列化：对象怎么在网络中传输?

下一篇　05 | 动态代理：面向接口编程，屏蔽RPC处理流程

精选留言 (17)

写留言

一步
2020-02-26

老师，我理解的IO多路复用，是应用线程一直再调用select，读取内核准备好数据的socket，所以应用线程是阻塞的，但是老师你文章中举的那个例子，不用应用(用户)打电话去询问的，而是内核(餐馆)打电话通知的，在这期间你还可以去干其他的事情，我感觉你这个案例是异步IO非阻塞的

展开

5　5

嬰
2020-03-01

IO多路复用分为select，poll和epoll，文中描述的应该是select的过程，nigix，redis等使

用的是epoll，所以光只看这个文章的话会比较迷惑，文中写的太粗了。
对于课后思考，目前很多的主流的需要通信的中间件都差不多都实现了零拷贝，如Kfaka，RocketMQ等。kafka的零拷贝是通过java.nio.channels.FileChannel中的transferTo方法来实现的，transferTo方法底层是基于操作系统的sendfile这个system call来实现的。

展开

2

川杰
2020-02-26

老师好，有句话不大理解：但它最大的优势在于，用户可以在一个线程内同时处理多个socket的IO请求；
1、这个线程指的是维护select的线程吗？
2、为什么用户可以在一个线程内处理多个socket的IO请求？我理解，这里的用户，应该指的是客户端调用方，那么多个socket，应该是指，其他客户端调用方发送过来的、且IO...

展开

2 2

阿卧
2020-03-01

阻塞IO：
1. 阻塞等待：多线程进行IO读取，需要阻塞等待
2. 内存两次拷贝：从设备（磁盘或者网络）拷贝到用户空间，再从用户空间拷贝到内核空间
IO多路复用...

展开

安排
2020-03-01

select监视一批socket文件描述符，当某个socket可读或者可写了，那么调用select的线程会被从阻塞中唤醒。这个过程的底层细节是啥啊？比如网卡中断收包，然后中断调用协议栈处理，一直到传输层，然后到socket，发现这个socket是被select监视的，然后将调用select的进程从阻塞队列放到可运行队列，最后在调度点被调度运行。这样理解对吗？

展开

redj
2020-02-29

老师说：多个网络连接的 IO 可以注册到一个复用器（select）上，当用户进程调用了 select，那么整个进程会被`阻塞`

既然是非阻塞 + IO多路复用，为什么这里会产生阻塞,阻塞的是内核的进程，还是用户的进程?
我记得应用进程在没有被通知的情况下，应用进程可以使用 CPU 做其他的事情，应用程...
展开

redj
2020-02-29

老师的话：多个网络连接的 IO 可以注册到一个复用器（select）上，当用户进程调用了 select，那么整个进程会被阻塞
展开

Dovelol
2020-02-29

老师好，IO多路复用也是为了解决C10K问题吧，能具体讲一下虚拟内存的作用吗，什么时候会用到，为什么要用。
展开

高源
2020-02-28

老师你讲的netty零拷贝不光做到在用户级别的，还有操作系统级别的，那老师如果我想理解它的怎么做的，因为它源代码看的迷糊，怎样把他实现的高技巧和方法运用到自己这块，灵活运用

yhh
2020-02-28

老师好，请教个问题，我看 零拷贝 虚拟内存 那张图，mmap是也能用于网络io吗?

💪😊
2020-02-27

关于网络这块，希望对epoll也进行一下深入对比啊！毕竟不管是nginx netty都使用它进行优化

川杰
2020-02-26

接上条，我重新思考了下；应该是这样，这里的事件驱动不是常规的客户端请求，而是指请求处理完后的状态变化，例如，当socket状态变为ready for reading时，相当于触发了新的请求，那么select会找到对应的handler进行调用；
所以，我刚才举得例子，即使只有一个客户端请求，也会即使的收到服务端的回复；因为这里所谓的event-driven不是指传统意义上的客户端请求的驱动，而是socket经handler...

展开

一步
2020-02-26

Redis的网络模型就是IO多路复用，但他是单线程的啊，老师文章中说到IO多路复用一般都是多线程进行实现的，这个怎么理解呢?

展开

1

小脚丫
2020-02-26

何老师您好，我是京东员工，看到您这章关于零拷贝的介绍，突然想起来个之前使用jmq的问题想请教您下，Jmq抛出了ERROR com.jd.jmq.common.network.netty.NettyTransport - netty channel exception 10.194.143.76:50088 io.netty.handler.codec.DecoderException: java.lang.IndexOutOfBoundsException这个异常，导致我们应用的堆内存里都是heapbytebuffer这个对象，占用了几个G，之后...

展开

作者回复: 私聊

3

Simon
2020-02-26

课后思考: RocketMQ使用mmap实现零拷贝

展开

作者回复: 这个也是

益
2020-02-26

请问.NET Core上有与Netty框架类似的网络通信框架吗?
请教老师，如果选择用gRPC开发接口服务，从底层原理角度，Java和.NET Core哪个更好呢?

1

张先生
2020-02-26

1.我的理解netty对应用层的零拷贝优化就是把做个tcp包做合并来减少频繁的cpu内核交互，但是cpu内核应该也有个大小限制吧?
2.零拷贝只是优化了服务器的开销，对于传输层并没有什么优化吧，因为传输层传输的包大小会受链路上路由可接收的包大小决定拆多少个包

展开

作者回复: netty零拷贝有多种支持，包括它提供的很多buffer。在进行文件网络传输，也可以进行零拷贝的优化。在网络传输过程中没有优化。