Linux 中的零拷贝技术，第 1 部分

概述

黄 晓晨 和 冯 瑞
2011 年 1 月 27 日发布

引言

传统的 Linux 操作系统的标准 I/O 接口是基于数据拷贝操作的，即 I/O 操作会导致数据在操作系
程序地址空间定义的缓冲区之间进行传输。这样做最大的好处是可以减少磁盘 I/O 的操作，因
作系统的高速缓冲存储器中，那么就不需要再进行实际的物理磁盘 I/O 操作。但是数据传输过
的 CPU 开销，限制了操作系统有效进行数据传输操作的能力。

零拷贝（zero-copy）这种技术可以有效地改善数据传输的性能，在内核驱动程序（比如网络
I/O 数据的时候，零拷贝技术可以在某种程度上减少甚至完全避免不必要 CPU 数据拷贝操作。
很多特征可以有效地实现零拷贝技术，但是因为存储体系结构非常复杂，而且网络协议栈有时
零拷贝技术有可能会产生很多负面的影响，甚至会导致零拷贝技术自身的优点完全丧失。

为什么需要零拷贝技术

如今，很多网络服务器都是基于客户端 - 服务器这一模型的。在这种模型中，客户端向服务器
需要响应客户端发出的请求，并为客户端提供它所需要的数据。随着网络服务的逐渐普及，vi
的计算机系统已经具备足够的能力去处理 video 这类应用程序对客户端所造成的重负荷，但是
video 这类应用程序引起的网络通信量就显得捉襟见肘了。而且，客户端的数量增长迅速，那
颈。而对于负荷很重的服务器来说，操作系统通常都是引起性能瓶颈的罪魁祸首。举个例子来
送”操作的系统调用发出时，操作系统通常都会将数据从应用程序地址空间的缓冲区拷贝到操

一般来说，客户端通过网络接口卡向服务器端发送请求，操作系统将这些客户端的请求传递给
程序会处理这些请求，请求处理完成以后，操作系统还需要将处理得到的结果通过网络适配器

下边这一小节会跟读者简单介绍一下传统的服务器是如何进行数据传输的，以及这种数据传输
造成服务器的性能损失。

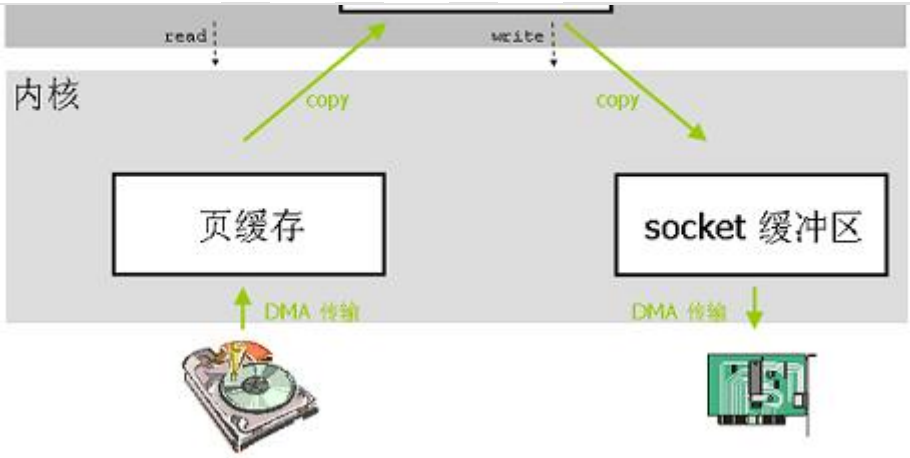
Linux 中传统服务器进行数据传输的流程

Linux 中传统的 I/O 操作是一种缓冲 I/O，I/O 过程中产生的数据传输通常需要在缓冲区中进
传输数据的时候，用户应用程序需要分配一块大小合适的缓冲区用来存放需要传输的数据。应
后把这块数据通过网络发送到接收端去。用户应用程序只是需要调用两个系统调用 read() 和 w
作，应用程序并不知晓在这个数据传输的过程中操作系统所做的数据拷贝操作。对于 Linux 操
验等各方面因素的考虑，操作系统内核会在处理数据传输的过程中进行多次拷贝操作。在某些
地降低数据传输的性能。

当应用程序需要访问某块数据的时候，操作系统内核会先检查这块数据是不是因为前一次对相
系统内核地址空间的缓冲区内，如果在内核缓冲区中找不到这块数据，Linux 操作系统内核会先
作系统内核的缓冲区里去。如果这个数据读取操作是由 DMA 完成的，那么在 DMA 进行数据读
行缓冲区管理，以及创建和处理 DMA ，除此之外，CPU 不需要再做更多的事情，DMA 执行完
统做进一步的处理。Linux 操作系统会根据 read() 系统调用指定的应用程序地址空间的地址，把
应用程序的地址空间中去，在接下来的处理过程中，操作系统需要将数据再一次从用户应用程
堆栈相关的内核缓冲区中去，这个过程也是需要占用 CPU 的。数据拷贝操作结束以后，数据会
去。在数据传输的过程中，应用程序可以先返回进而执行其他的操作。之后，在调用 write() 系
区中的数据内容可以被安全的丢弃或者更改，因为操作系统已经在内核缓冲区中保留了一份数
上之后，这份数据拷贝就可以被丢弃。

从上面的描述可以看出，在这种传统的数据传输过程中，数据至少发生了四次拷贝操作，即便
讯，CPU 仍然需要访问数据两次。在 read() 读数据的过程中，数据并不是直接来自于硬盘，而
层。在 write() 写数据的过程中，为了和要传输的数据包的大小相吻合，数据必须要先被分割成
要进行数据校验和操作。

图 1. 传统使用 read 和 write 系统调用的数据传输



零拷贝（zero copy）技术概述

什么是零拷贝?

简单一点来说，零拷贝就是一种避免 CPU 将数据从一块存储拷贝到另外一块存储的技术。针对
系统以及网络协议堆栈而出现的各种零拷贝技术极大地提升了特定应用程序的性能，并且使得
系统资源。这种性能的提升就是通过在数据拷贝进行的同时，允许 CPU 执行其他的任务来实现
和共享总线操作的次数，消除传输数据在存储器之间不必要的中间拷贝次数，从而有效地提高
减少了用户应用程序地址空间和操作系统内核地址空间之间因为上下文切换而带来的开销。进
简单的任务，从操作系统的角度来说，如果 CPU 一直被占用着去执行这项简单的任务，那么这
比较简单的系统部件可以代劳这件事情，从而使得 CPU 解脱出来可以做别的事情，那么系统资
述，零拷贝技术的目标可以概括如下：

避免数据拷贝

- 避免操作系统内核缓冲区之间进行数据拷贝操作。
- 避免操作系统内核和用户应用程序地址空间这两者之间进行数据拷贝操作。
- 用户应用程序可以避开操作系统直接访问硬件存储。
- 数据传输尽量让 DMA 来做。

将多种操作结合在一起

- 避免不必要的系统调用和上下文切换。
- 需要拷贝的数据可以先被缓存起来。
- 对数据进行处理尽量让硬件来做。

别的事情，这就产生了性能瓶颈，限制了通讯速率，从而降低了网络链接的能力。一般来说，数据。举例来说，一个 1 GHz 的处理器可以对 1Gbit/s 的网络链接进行传统的数据拷贝操作，么对于相同的处理器来说，零拷贝技术就变得非常重要了。对于超过 1 Gbit/s 的网络链接来说及大型的商业数据中心中都有所应用。然而，随着信息技术的发展，1 Gbit/s，10 Gbit/s 以及那么零拷贝技术也会变得越来越普及，这是因为网络链接的处理能力比 CPU 的处理能力的增长传统的操作系统或者通信协议，这就限制了数据传输性能。零拷贝技术通过减少数据拷贝次数序和网络之间提供更快的数据传输方法，从而可以有效地降低通信延迟，提高网络吞吐率。零设备高速网络接口的主要技术之一。

现代的 CPU 和存储体系结构提供了很多相关的功能来减少或避免 I/O 操作过程中产生的不必要 CPU 和存储体系结构的这种优势经常被过高估计。存储体系结构的复杂性以及网络协议中必需甚至会导致零拷贝这种技术的优点完全丧失。在下一章中，我们会介绍几种 Linux 操作系统中它们的实现方法，并对它们的弱点进行分析。

零拷贝技术分类

零拷贝技术的发展很多样化，现有的零拷贝技术种类也非常多，而当前并没有一个适合于所有 Linux 来说，现存的零拷贝技术也比较多，这些零拷贝技术大部分存在于不同的 Linux 内核版本核版本间得到了很大的发展或者已经渐渐被新的技术所代替。本文针对这些零拷贝技术所适用括起来，Linux 中的零拷贝技术主要有下面这几种：

- 直接 I/O：对于这种数据传输方式来说，应用程序可以直接访问硬件存储，操作系统内核只针对的是操作系统内核并不需要对数据进行直接处理的情况，数据可以在应用程序地址空间输，完全不需要 Linux 操作系统内核提供的页缓存的支持。
- 在数据传输的过程中，避免数据在操作系统内核地址空间的缓冲区和用户应用程序地址空间候，应用程序在数据进行传输的过程中不需要对数据进行访问，那么，将数据从 Linux 的页就可以完全避免，传输的数据在页缓存中就可以得到处理。在某些特殊的情况下，这种零拷 Linux 中提供类似的系统调用主要有 mmap()，sendfile() 以及 splice()。
- 对数据在 Linux 的页缓存和用户进程的缓冲区之间的传输过程进行优化。该零拷贝技术侧重缓冲区和操作系统的页缓存之间的拷贝操作。这种方法延续了传统的通信方式，但是更加灵要利用了写时复制技术。

前两类方法的目的主要是为了避免应用程序地址空间和操作系统内核地址空间这两者之间的缓通常适用在某些特殊的情况下，比如要传送的数据不需要经过操作系统内核的处理或者不需要则继承了传统的应用程序地址空间和操作系统内核地址空间之间数据传输的概念，进而针对数硬件和软件之间的数据传输可以通过使用 DMA 来进行，DMA 进行数据传输的过程中几乎不 CPU 解放出来去做更多其他的事情，但是当数据需要在用户地址空间的缓冲区和 Linux 操

总结

本系列文章介绍了 Linux 中的零拷贝技术，本文是其中的第一部分，介绍了零拷贝技术的基本
种技术以及简要概述了 Linux 中都存在哪些零拷贝技术这样一些基本背景知识。我们将在本系
本文提到的 Linux 中的几种零拷贝技术。

相关主题

- Zero Copy I: User-Mode Perspective, Linux Journal (2003), http://www.linuxjournal.com/
系统中存在的零拷贝技术。
- 在 http://www.ibm.com/developerworks/cn/aix/library/au-tcpsystemcalls/这篇文章中可
绍。
- 关于 Linux 中直接 I/O 技术的设计与实现请参考 developerWorks 上的文章 ”Linux 中直接 I
- http://lwn.net/Articles/28548/上可以找到 Linux 上零拷贝技术中关于用户地址空间访问技
- http://articles.techrepublic.com.com/5100-10878_11-1044112.html?tag=content;leftC
sendfile() 优化数据传输的介绍。
- 关于 splice() 系统调用的介绍，可以参考 http://lwn.net/Articles/178199/。
- http://en.scientificcommons.org/43480276这篇文章介绍了如何通过在操作系统内核中直
CPU 的可用性，并详细描述了 splice 系统调用如何在用户应用程序不参与的情况下进行异
- 在 Understanding the Linux Kernel(3rd Edition) 中，有关于 Linux 内核的实现。
- http://pages.cs.wisc.edu/~cao/cs736/slides/cs736-class18.ps这篇文章提出并详细介绍
机制，以及基于共享存储的数据传输机制。
- http://www.cs.inf.ethz.ch/group/stricker/sada/archive/chihaia.pdf，这篇文章列举了一些
测试系统上实现了 fbufs 技术。
- 在 developerWorks Linux 专区寻找为 Linux 开发人员（包括 Linux 新手入门）准备的更多
章和教程。
- 在 developerWorks 上查阅所有 Linux 技巧和 Linux 教程。
- 随时关注 developerWorks 技术活动和 网络广播。