特立独行MVP

【第三章：BAT等名企面试真题解析8讲】第13节：百度面试真题解析之Zero Copy技术 已购

来自【《收割BAT：C++校招学习路线总结》】 | 67 浏览 | 0 回复 | 2020-02-20

特立独行MVP 专栏作者 +关注

前言

在我秋招面试过程当中，关于Zero Copy的问题一共出现了3次。对于一个程序员来说，当一个问题出现了3次那么就需要重视和总结，事不过三。本节我会对Zero Copy做一个介绍，并结合百度面试真题进行解析。

系统级IO

要理解Zero Copy技术的出现，首先需要知道什么是系统级IO。Linux系统级IO分为：

1.标准IO库

2.IO系统调用

3.网络IO库

IO系统调用：

Linux标准访问文件方式是通过两个系统调用实现的：read()和write()，这两个系统调用在**用户态都是没有缓冲**。当用户进程使用read 和 write 读写Linux的文件时,进程会从用户态进入内核态，通过I/O操作读取文件中的数据。

read ():

```
1 ssize_t read(int fd, void * buf, size_t count);
```

read()会把参数fd所指的文件传送count 个字节到buf 指针所指的内存中。

write():

```
1 ssize_t write (int fd, const void * buf, size_t count);
```

write()会把参数buf所指的内存写入count个字节到参数放到所指的文件内。

举例来说，要写入数据到文件上时，内核先将数据写入到内核中所设的缓冲当中；假如这个缓冲储存器的长度是100字节，调用系统函数write时，假设每次要写入的数据的长度为10个字节，那么你几要调用10次write函数才能将内核缓冲区写满；此时数据还是在缓冲区，并没有写入到磁盘，缓冲

标准IO库：

标准IO库是基于IO系统调用实现的，优化了对系统调用的使用方式。引入标准IO库有以下几个原因：

1.因为IO系统调用的使用方式非常底层，需要指定读写的count以及buf，使用比较麻烦，所以标准IO库对IO系统调用进行封装。

2.因为read 和 write 等底层系统调用需要在**用户态**和**内核态**之间切换，如果每次读写的数据很少，那么切换带来的开销将大大降低IO的效率，所以标准IO库在用户态也引入了缓冲机制，提升了性能。

3.IO系统调用在不同的操作系统之间是不能通用的，但是标准IO库在不同的平台几乎是一致的，这就增强了可移植性。

常见的标准IO库函数：

fopen、fclose、fwrite、fread、ffulsh、fseek等等。

Zero-Copy

零拷贝技术是指计算机执行操作时，CPU不需要先将数据从某处内存复制到另一个特定区域。这种技术通常用于通过网络传输文件时**节省CPU周期和内存带宽**。举例来说，如果要从磁盘当中读取一个文件并通过网络发送它，传统方式下每个读/写周期都需要复制两次数据和切换两次上下文，而**数据的复制都需要依靠CPU**。

1.Linux 2.1内核引入sendfile函数：

sendfile通过一次系统调用完成了文件的传送，通过sendfile发送文件只需要一次系统调用，当调用sendfile时数据的拷贝路径如下：

第一次拷贝：将数据从磁盘读取到内核缓冲区中；

第二次拷贝：将数据从内核缓冲区拷贝到socket buffer中；

第三次拷贝：将数据从socket buffer拷贝到网卡设备中发送；

2.Linux2.4 内核对sendfile做了进一步的改进：

改进后的数据拷贝处理路径如下：

第一次拷贝：将文件从磁盘拷贝到内核缓冲区中，不再将内核缓冲区的数据拷贝到socket buffer，而是向socket buffer中写入当前要发送的数据在内核缓冲区中的位置和偏移量；

特立独行MVP

百度面试真题

现在有一个用户需要**读取磁盘文件上的内容然后将其通过网络**发送出去，假设使用IO系统调用read/write

1.这个过程当中，数据经过几次拷贝？

答：在这个过程中经历了4次数据copy的过程，路径如下：

第一次数据拷贝：调用read时，文件从磁盘拷贝到了内核缓冲区；

第二次数据拷贝：数据从内核缓冲区拷贝到用户内存缓冲区；

第三次数据拷贝：调用write时，将用户内存缓冲区的数据内容拷贝到内核模式下的socket的buffer中；

第四次数据拷贝：最后将内核模式下的socket buffer的数据拷贝到网卡设备中；

2.这个过程当中，出现了几次用户态和内核态的切换？

答：在这个过程中经历了4次用户态和内核态的切换：

第一次切换：调用read时用户态切换到内核态；

第二次切换：read调用返回，内核态切换回用户态；

第三次切换：调用write时用户态切换到内核态；

第四次切换：write返回时内核态切换到用户态；

3.这个过程当中，需要几次系统调用？

答：这个过程当中经历了2次系统调用，分别是read和write系统调用。

4.有什么方法可以优化这个过程？

答：使用zero copy，避免数据在用户态的拷贝以及减少系统调用次数。

结论

特立独行MVP

上下文的切换。对于Zero-copy需要记住以下2点：

1. Zero-copy可以将读取磁盘文件网络传输的上下文切换的次数从4次降低到2次；数据拷贝次数从4次降低到2次；
2. Zero-copy是针对内核来说，数据在内核模式下是无拷贝过程，并不是指整个过程数据没有拷贝。

结语

感谢阅读，如果文中有任何错误欢迎你给我留言指出，也欢迎分享给更多的朋友一起阅读。

特立独行MVP

牛客博客，记录你的成长

关于博客 | 意见反馈 | 免责声明 | 牛客网首页