

Java NIO之NIO与传统IO的区别



六尺帐篷 (/u/f8e9b1c246f1)

2017.08.01 10:55 字数 1599 阅读 129 评论 0 喜欢 5

(/u/f8e9b1c246f1)

[编辑文章 \(/writer#/notebooks/14906121/notes/15256887\)](#)

- IO
- NIO
- 小结

I/O

I/O？或者输入/输出？指的是计算机与外部世界或者一个程序与计算机的其余部分的之间的接口。它对于任何计算机系统都非常关键，因而所有 I/O 的主体实际上是内置在操作系统中的。单独的程序一般是让系统为它们完成大部分的工作。

在 Java 编程中，直到最近一直使用 流 的方式完成 I/O。所有 I/O 都被视为单个的字节的移动，通过一个称为 Stream 的对象一次移动一个字节。流 I/O 用于与外部世界接触。它也在内部使用，用于将对象转换为字节，然后再转换回对象。

传统流IO的好处是使用简单，将底层的机制都抽象成流，但缺点就是性能不足。而且IO的各种流是阻塞的。这意味着，当一个线程调用read() 或 write()时，该线程被阻塞，直到有一些数据被读取，或数据完全写入。该线程在此期间不能再干任何事情了。

以socket.read()为例子：

传统的BIO里面socket.read()，如果TCP RecvBuffer里没有数据，函数会一直阻塞，直到收到数据，返回读到的数据。

NIO

为什么要使用 NIO？

NIO 的创建目的是为了让 Java 程序员可以实现高速 I/O 而无需编写自定义的本机代码。NIO 将最耗时的 I/O 操作(即填充和提取缓冲区)转移回操作系统，因而可以极大地提高速度。

流与块的比较

原来的 I/O 库(在 java.io.*中) 与 NIO 最重要的区别是数据打包和传输的方式。正如前面提到的，原来的 I/O 以流的方式处理数据，而 NIO 以块的方式处理数据。

面向流的 I/O 系统一次一个字节地处理数据。一个输入流产生一个字节的数据，一个输出流消费一个字节的数据。为流式数据创建过滤器非常容易。链接几个过滤器，以便每个过滤器只负责单个复杂处理机制的一部分，这样也是相对简单的。不利的一面是，面向流的 I/O 通常相当慢。

一个面向块的 I/O 系统以块的形式处理数据。每一个操作都在一步中产生或者消费一个数据块。按块处理数据比按(流式的)字节处理数据要快得多。但是面向块的 I/O 缺少一些面向流的 I/O 所具有优雅性和简单性。

NIO的buffer机制

NIO性能的优势就来源于缓冲的机制，不管是读或者写都需要以块的形式写入到缓冲区中。NIO实际上让我们对IO的操作更接近于操作系统的实际过程。

所有的系统I/O都分为两个阶段：等待就绪和操作。举例来说，读函数，分为等待系统可读和真正的读；同理，写函数分为等待网卡可以写和真正的写。



以socket为例：
先从应用层获取数据到内核的缓冲区，然后再从内核的缓冲区复制到进程的缓冲区。所以实际上底层的机制也是不断利用缓冲区来读写数据的。即使传统IO抽象成了从流直接读取数据，但本质上也依然是利用缓冲区来读取和写入数据。
所以，为了更好的理解nio，我们就需要知道IO的底层机制，这样对我们将来理解channel和buffer就打下了基础。这里简单提一下，我们可以把buffer就理解为内核缓冲区，所以不论读写，自然都要经过这个区域，读的话，先从设备读取数据到内核，再读到进程缓冲区，写的话，先从进程缓冲区写到内核，再从内核写回设备。

NIO的非阻塞机制

NIO的非阻塞模式，使一个线程从某通道发送请求读取数据，但是它仅能得到目前可用的数据，如果目前没有数据可用时，就什么也不会获取。而不是保持线程阻塞，所以直至数据变的可以读取之前，该线程可以继续做其他的事情。非阻塞写也是如此。一个线程请求写入一些数据到某通道，但不需要等待它完全写入，这个线程同时可以去做别的事情。线程通常将非阻塞IO的空闲时间用于在其它通道上执行IO操作，所以一个单独的线程现在可以管理多个输入和输出通道（channel）。

下图是几种常见I/O模型的对比：



以socket.read()为例子：

传统的BIO里面socket.read()，如果TCP RecvBuffer里没有数据，函数会一直阻塞，直到收到数据，返回读到的数据。

对于NIO，如果TCP RecvBuffer有数据，就把数据从网卡读到内存，并且返回给用户；反之则直接返回0，永远不会阻塞。所以我们可以NIO实现同时监听多个IO通道，然后不断的轮询寻找可以读写的设备。

NIO的IO模型可以理解为是IO多路复用模型和非阻塞模型，同时还有事件驱动模型。这里需要知道一点，就是IO多路复用是一定要实现非阻塞的。

小结

NIO相对于IO流的优势：

- 非阻塞
- buffer机制

^

📄

🔖

🔗

- 流替代块

参考：

- <https://tech.meituan.com/nio.html> (<https://tech.meituan.com/nio.html>)
- <http://www.importnew.com/19816.html> (<http://www.importnew.com/19816.html>)
- <https://www.ibm.com/developerworks/cn/education/java/j-nio/j-nio.html>
(<https://www.ibm.com/developerworks/cn/education/java/j-nio/j-nio.html>)

📖 Java NIO (/nb/14906121)

© 著作权归作者所有



六尺帐篷 (/u/f8e9b1c246f1)

写了 245418 字，被 15240 人关注，获得了 1429 个喜欢

(/u/f8e9b1c246f1)

如果觉得我的文章对您有用，请随意赞赏。您的支持将鼓励我继续创作！

赞赏支持


 喜欢

5



更多分享

(<http://cwb.assets.jianshu.io/notes/images/152568f>)




写下你的评论...


评论 关闭评论

智慧如你，不想发表一点想法咩~

被以下专题收入，发现更多相似内容

🔧 投稿管理


- + 收入我的专题
- 


Android知识 (/c/3fde3b545a35?utm_source=desktop&utm_medium=notes-included-collection)
- 


Android... (/c/5139d555c94d?utm_source=desktop&utm_medium=notes-included-collection)
- 


Android开发 (/c/d1591c322c89?utm_source=desktop&utm_medium=notes-included-collection)
- 


Android... (/c/58b4c20abf2f?utm_source=desktop&utm_medium=notes-included-collection)











程序员 (/c/NEt52a?utm_source=desktop&utm_medium=notes-included-collection)

