

一文读懂 mmap 原理



JaydenLie LV.3

2021年04月27日 19:36 · 阅读 6600

关注

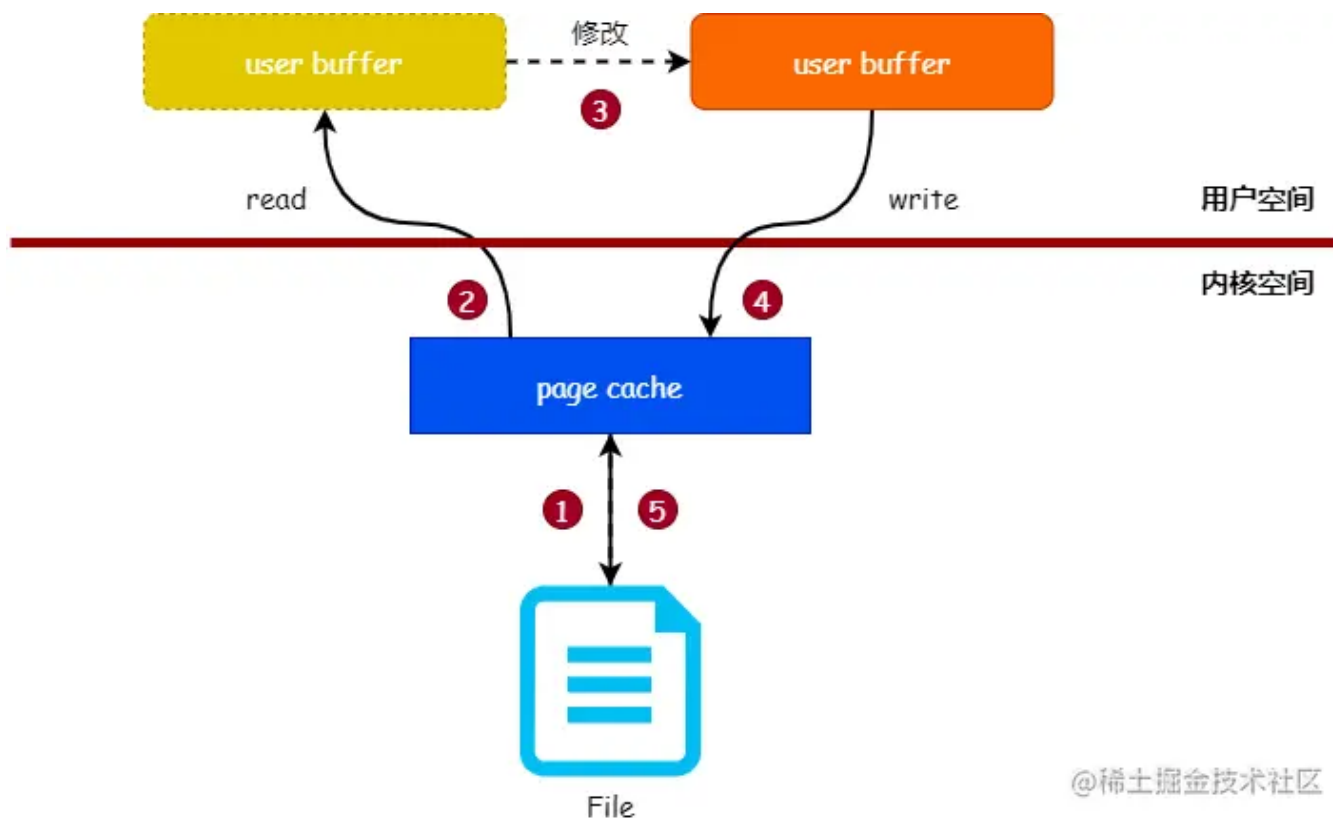
在《[一文看懂零拷贝技术](#)》中我们介绍了 [零拷贝技术](#) 的原理，而且我们知道 [mmap](#) 也是零拷贝技术的一种实现。在本文中，我们主要介绍 [mmap](#) 的原理。

一、传统的读写文件

一般来说，修改一个文件的内容需要如下3个步骤：

- 把文件内容读入到内存中。
- 修改内存中的内容。
- 把内存的数据写入到文件中。

过程如图 1 所示：



@稀土掘金技术社区

如果使用代码来实现上面的过程，代码如下：

```
read(fd, buf, 1024); // 读取文件的内容到buf
...                // 修改buf的内容
write(fd, buf, 1024); // 把buf的内容写入到文件
```

c 复制代码

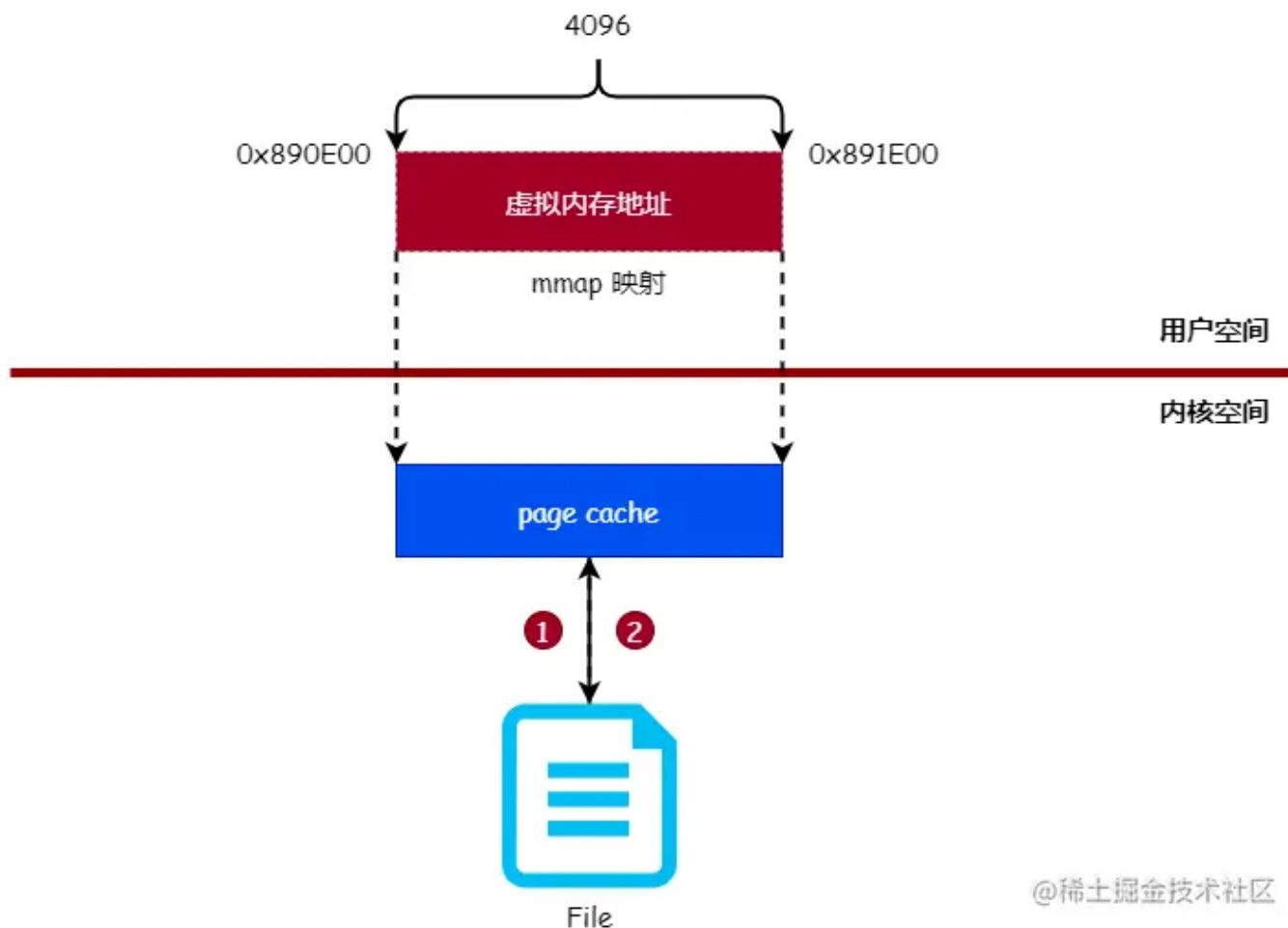
从图 1 中可以看出，**页缓存(page cache)** 是读写文件时的中间层，内核使用 **页缓存** 与文件的数据块关联起来。所以应用程序读写文件时，实际操作的是 **页缓存**。

二、使用 mmap 读写文件

从传统读写文件的过程中，我们可以发现有个地方可以优化：如果可以直接在用户空间读写 **页缓存**，那么就可以免去将 **页缓存** 的数据复制到用户空间缓冲区的过程。

那么，有没有这样的技术能实现上面所说的方式呢？答案是肯定的，就是 **mmap**。

拟内存地址进行读写操作就如同对文件进行读写操作一样。原理如图 2 所示：



前面我们介绍过，读写文件都需要经过 **页缓存**，所以 **mmap** 映射的正是文件的 **页缓存**，而非磁盘中的文件本身。由于 **mmap** 映射的是文件的 **页缓存**，所以就涉及到同步的问题，即 **页缓存** 会在什么时候把数据同步到磁盘。

Linux 内核并不会主动把 **mmap** 映射的 **页缓存** 同步到磁盘，而是需要用户主动触发。同步 **mmap** 映射的内存到磁盘有 4 个时机：

- 调用 **msync** 函数主动进行数据同步（主动）。
- 调用 **munmap** 函数对文件进行解除映射关系时（主动）。
- 进程退出时（被动）。
- 系统关机时（被动）。

下面我们介绍一下如何使用 `mmap` , `mmap` 函数的原型如下:

```
void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);
```

[c 复制代码](#)

下面介绍一下 `mmap` 函数的各个参数作用:

- `addr` : 指定映射的虚拟内存地址, 可以设置为 `NULL`, 让 Linux 内核自动选择合适的虚拟内存地址。
- `length` : 映射的长度。
- `prot` : 映射内存的保护模式, 可选值如下:
 - `PROT_EXEC` : 可以被执行。
 - 2. `PROT_READ` : 可以被读取。
 - 3. `PROT_WRITE` : 可以被写入。
 - 4. `PROT_NONE` : 不可访问。
- `flags` : 指定映射的类型, 常用的可选值如下:
 - `MAP_FIXED` : 使用指定的起始虚拟内存地址进行映射。
 - 2. `MAP_SHARED` : 与其它所有映射到这个文件的进程共享映射空间 (可实现共享内存) 。
 - 3. `MAP_PRIVATE` : 建立一个写时复制 (Copy on Write) 的私有映射空间。
 - 4. `MAP_LOCKED` : 锁定映射区的页面, 从而防止页面被交换出内存。
 - 5. ...
- `fd` : 进行映射的文件句柄。
- `offset` : 文件偏移量 (从文件的何处开始映射) 。

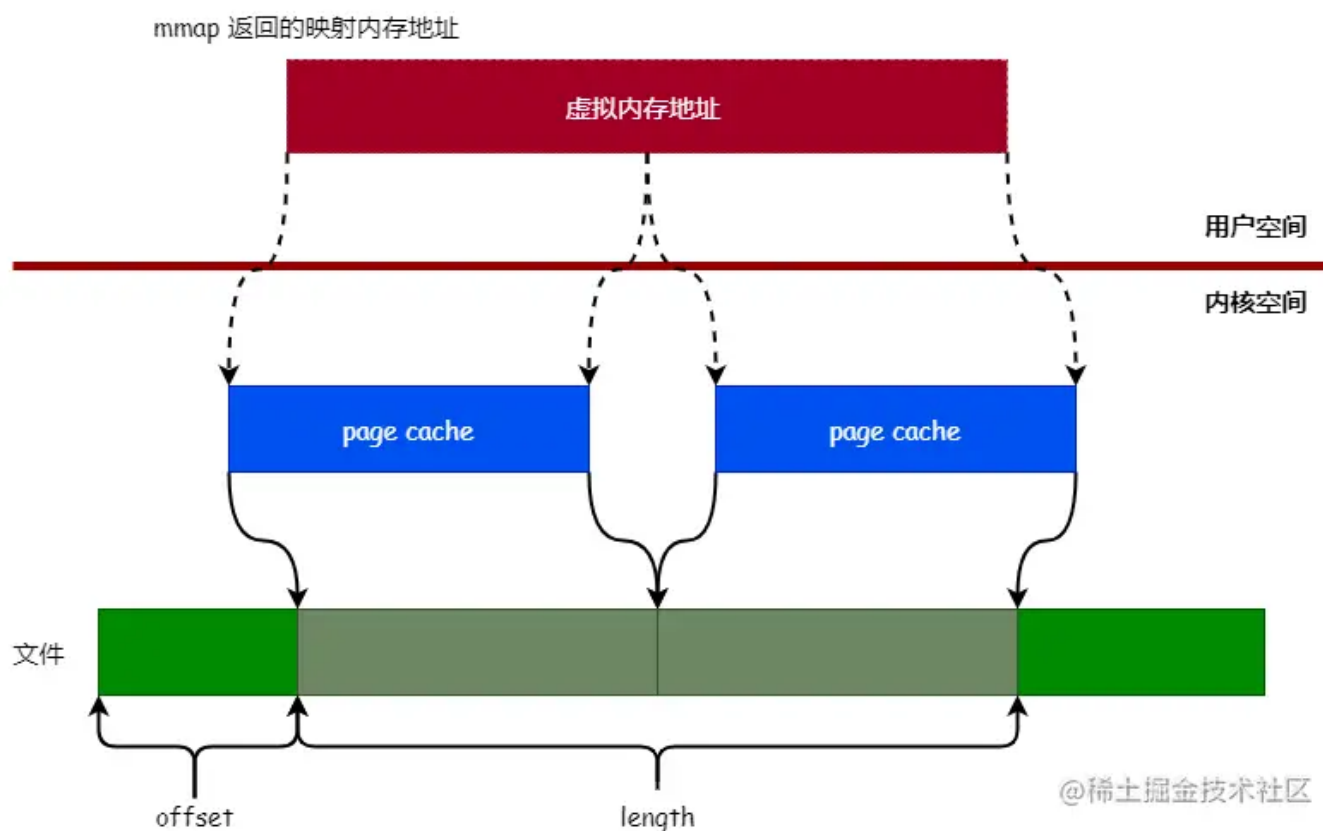
介绍完 `mmap` 函数的原型后, 我们现在通过一个简单的例子介绍如何使用 `mmap` :

```
void *addr = mmap(NULL, 8192, PROT_WRITE, MAP_SHARED, fd, 4096); // 对文件进行映射
```

在上面例子中，我们先通过 `open` 函数以可读写的方式打开文件，然后通过 `mmap` 函数对文件进行映射，映射的方式如下：

- `addr` 参数设置为 `NULL`，表示让操作系统自动选择合适的虚拟内存地址进行映射。
- `length` 参数设置为 `8192` 表示映射的区域为 2 个内存页的大小（一个内存页的大小为 4 KB）。
- `prot` 参数设置为 `PROT_WRITE` 表示映射的内存区为可读写。
- `flags` 参数设置为 `MAP_SHARED` 表示共享映射区。
- `fd` 参数设置打开的文件句柄。
- `offset` 参数设置为 `4096` 表示从文件的 4096 处开始映射。

`mmap` 函数会返回映射后的内存地址，我们可以通过此内存地址对文件进行读写操作。我们通过图 3 展示上面例子在内核中的结构：



四、总结

本文主要介绍了 `mmap` 的原理和使用方式，通过本文我们可以知道，使用 `mmap` 对文件进行读写操作时可以减少内存拷贝的次数，并且可以减少系统调用的次数，从而提高对读写文件操作的效率。

由于内核不会主动同步 `mmap` 所映射的内存区中的数据，所以在某些特殊的场景下可能会出现数据丢失的情况（如断电）。为了避免数据丢失，在使用 `mmap` 的时候可以在适当时主动调用 `msync` 函数来同步映射内存区的数据。

分类： 后端 标签： [Linux](#)

安装掘金浏览器插件

多内容聚合浏览、多引擎快捷搜索、多工具便捷提效、多模式随心畅享，你想要的，这里都有！

[前往安装](#)


评论




输入评论（Enter换行，Ctrl + Enter发送）



全部评论 2


[最新](#) [最热](#)



- 




爱你真是太好了 

牛啊 学到了

 点赞  回复

8月前
- 

大茄   掘金社区

 15  2  收藏

11日前