

Linux IO漫谈

本文为原创，转载请注明：
<http://www.cnblogs.com/gistao/>
Background

IO可能是我们接触最频繁的系统调用，比如printf到终端，send content到对端，而今天要讨论的仅是Linux平台下访问本机存储设备相关的IO。如果你对IO相关api的优缺点门清，可以忽略这个随笔啦。

read

read的过程大致如下：

- 1. 用户malloc出一块内存，然后陷入内核。
- 2. 内核从磁盘读取内容拷贝到cache。
- 3. 内核将内容拷贝到用户内存。

缺点比较明显，需要两次拷贝，拷贝是非常耗cpu的。

O_DIRECT

open函数的参数里有这样一个flag，意思是说不需要内核做cache了，内核直接把数据memcpy给用户就好了，这样的优点是可以减少memcpy。但缺点也很明显，没有cache了。

pread

多线程代码并发的访问同一文件是常见的，示例代码如下：

```
pthread_mutex_lock (mutex);

lseek (SEEK_SET+1024);
read (buf);

pthread_mutex_unlock (mutex);
```

这里锁的意义是防治文件指针被其他线程seek走，导致本次read错乱，如何避免掉这个锁呢，就是pread，此函数和read的功能一样，但增加了一个要读取的offset参数，这样就不需要我们显示的加锁了，也许你会联想到strtok和strtok_r。代码改动如下：

```
pread (fd, buf, offset);
```

readahead

有时又会遇到下边代码的场景

```
while (n < max) {
    read (buf[i]);
    on_handle (buf[i]);
}
```

公告

昵称： gisTao
园龄： 6年11个月
粉丝： 3
关注： 0
[+加关注](#)

导航

[博客园](#)
[首页](#)
[新随笔](#)
[联系](#)
[订阅](#) 
[管理](#)

< 2019年12月 >						
日	一	二	三	四	五	六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

统计

随笔 - 14
文章 - 0
评论 - 4
引用 - 0

搜索

常用链接


[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

我的标签

[原创](#)(11)
[科普](#)(2)
[面试](#)(1)
[async](#)(1)
[golang](#)(1)
[leveldb](#)(1)
[non-blocking](#)(1)
[pb](#)(1)
[redigo](#)(1)
[redis](#)(1)
[更多](#)


随笔分类

read函数是阻塞的，所以执行时间就等于max * time，即串行执行。可不可以非阻塞，就是readahead，此函数意思是在read之前，非阻塞的通知内核一下我要读的内容，内核会并发的预读这些内容进cache，当后续进行read时会大大的减少时间，代码修改成这样：



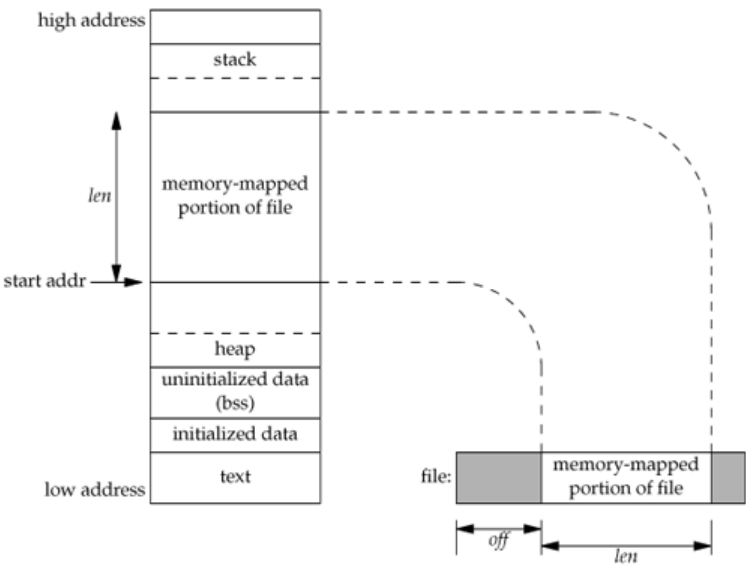
```
while (n < max) {
    readahead (offset);
}

while (n < max) {
    read (buf[i]);
    on_handle (buf[i]);
}
```



mmap

mmap的详细原理不在此讨论，见图



简单来说，是将内核空间映射到了用户空间，这样相比read函数来说减少了一次memcpy。

而mmap比O_DIRECT也有个好处是它确实利用了cache，可是对于普通的read来说，它利用的又不够充分，因为并不是每次访问都需要内核参与。不过有个补足办法就是用readahead，它可以传递你的意图(利用cache)给内核，从而避免了缺点。示例代码如下：

```
readahead (fd , offset);
ptr = mmap (fd);
a = ptr[offset];
```

aio

从以上可以看出来，IO模式是从阻塞提升到了非阻塞，性能优化围绕着cache和memcpy，那有没有一种非阻塞的，有cache的，最少memcpy的，这些都符合的技术，那应该就是aio了吧，但Linux的aio到现在也没有一个很好的实现，也许过于复杂吧，反过来看下Window平台，这都真不是事。下图是aio模型：

[hhvm\(2\)](#)
[并行编程\(3\)](#)
[网络\(6\)](#)

随笔档案

- [2017年8月\(1\)](#)
- [2016年8月\(1\)](#)
- [2015年7月\(1\)](#)
- [2015年6月\(1\)](#)
- [2015年5月\(1\)](#)
- [2015年4月\(7\)](#)
- [2015年3月\(1\)](#)
- [2013年1月\(1\)](#)

最新评论

- [1. Re: folly::AtomicHashMap源码分析\(二\)](#)
numEntries_为thread_cache_int类型，这个类的大致思想是可以配置一个cache_size，那么访问这个numEntries_对象的所有线程的局部变量++到cache_size后...
--hcarry2672

阅读排行榜

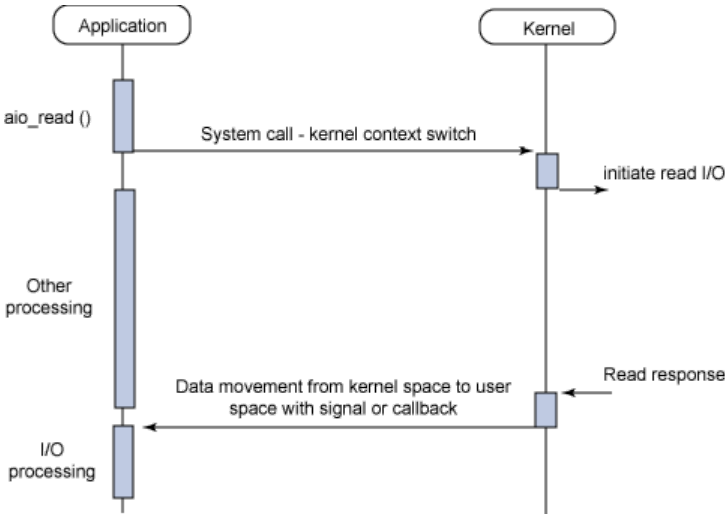
- [1. folly::AtomicHashMap源码分析\(一\)\(1857\)](#)
- [2. 海量结构化日志分析系统\(1397\)](#)
- [3. folly::AtomicHashMap源码分析\(二\)\(1096\)](#)
- [4. 一个共享内存hash\(800\)](#)
- [5. Linux IO漫谈\(699\)](#)

评论排行榜

- [1. folly::AtomicHashMap源码分析\(二\)\(1\)](#)

推荐排行榜

- [1. folly::AtomicHashMap源码分析\(一\)\(1\)](#)
- [2. folly::AtomicHashMap源码分析\(二\)\(1\)](#)



Final

有了mmap + ahead, aio真的还那么重要吗? 真的很重要, 没有真正的异步, 就没有真正的**并行编程**, 不能实现真正的async和await语法糖, 比如一个异步方法要求10ms必须返回, 而系统调用就得50ms完成, 怎么可能是真正的**async**. 但是如果不吹毛求疵, 或追求极致的话, 目前确实够用了, 通过以上总结, 希望能在IO编程时给你些选择素材。

分类: [网络](#)

标签: [原创](#), [科普](#)

好文要顶

关注我

收藏该文

gisTao

关注 - 0

粉丝 - 3

+ 加关注

0

0

« 上一篇: [躲不开的多线程](#)
» 下一篇: [folly::AtomicHashMap源码分析\(一\)](#)

posted on 2015-05-06 17:26 [gisTao](#) 阅读(699) 评论(0) [编辑](#) [收藏](#)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论, 请 [登录](#) 或 [注册](#), [访问](#) 网站首页。

- 【推荐】超50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】腾讯云热门云产品限时秒杀, 爆款1核2G云服务器99元/年!
- 【推荐】阿里云双11返场来袭, 热门产品低至一折等你来抢!
- 【推荐】百度智能云岁末感恩季, 明星产品低至1元新老用户畅享
- 【活动】京东云服务器_云主机低于1折, 低价高性能产品备战双11
- 【活动】ECUG For Future 技术者的年度盛会 (杭州, 1月4-5日)

- 相关博文:
- [Linux的IO系统常用系统调用及分析](#)
 - [高性能网络IO模型](#)
 - [漫谈linux文件IO](#)

- [漫谈NIO\(1\)之计算机IO实现](#)
- [Linux的IO模型讲解](#)
- » [更多推荐...](#)

[35个面试详解](#), [170道挑战题](#), [1460个精彩问答](#) | [最全Java工程师面试宝典](#)

最新 IT 新闻:

- [微信电脑版有多难用，你们真的没感觉吗](#)
- [滴滴总裁柳青：平安夜饮酒乘客可能接到我的"电话"](#)
- [1344天，企业微信在腾讯ToB大战略中找到了位置](#)
- [怪兽充电完成5亿元C轮融资，软银愿景基金领投](#)
- [国产CPU新里程碑！龙芯3号新一代发布，性能追赶AMD](#)
- » [更多新闻...](#)

Powered by: [博客园](#) Copyright © 2019 gisTao
Powered by .NET Core 3.1.0 on Linux