

## 32 | 答疑（四）：阻塞、非阻塞 I/O 与同步、异步 I/O 的区别和联系

2019-02-01 倪朋飞

Linux性能优化实战

[进入课程 >](#)



讲述：冯永吉

时长 07:03 大小 6.47M



你好，我是倪朋飞。

专栏更新至今，四大基础模块的第三个模块——文件系统和磁盘 I/O 篇，我们就已经学完了。很开心你还没有掉队，仍然在积极学习思考和实践操作，并且热情地留言与讨论。

今天是性能优化的第四期。照例，我从 I/O 模块的留言中摘出了一些典型问题，作为今天的答疑内容，集中回复。同样的，为了便于你学习理解，它们并不是严格按照文章顺序排列的。

每个问题，我都附上了留言区提问的截屏。如果你需要回顾内容原文，可以扫描每个问题右下方的二维码查看。

## 问题 1：阻塞、非阻塞 I/O 与同步、异步 I/O 的区别和联系

石维康	
写于 2019/01/11	阻塞 I/O 和非阻塞 I/O 的概念和同步和异步 I/O 的区别是什么？
<div>引自：Linux 性能优化实战 23   基础篇：Linux 文件系统是怎么工作的？</div> <div>识别二维码打开原文 「极客时间」App</div> 	

在[文件系统的工作原理](#)篇中，我曾经介绍了阻塞、非阻塞 I/O 以及同步、异步 I/O 的含义，这里我们再简单回顾一下。

首先我们来看阻塞和非阻塞 I/O。根据应用程序是否阻塞自身运行，可以把 I/O 分为阻塞 I/O 和非阻塞 I/O。

所谓阻塞 I/O，是指应用程序在执行 I/O 操作后，如果没有获得响应，就会阻塞当前线程，不能执行其他任务。

所谓非阻塞 I/O，是指应用程序在执行 I/O 操作后，不会阻塞当前的线程，可以继续执行其他的任务。

再来看同步 I/O 和异步 I/O。根据 I/O 响应的通知方式的不同，可以把文件 I/O 分为同步 I/O 和异步 I/O。

所谓同步 I/O，是指收到 I/O 请求后，系统不会立刻响应应用程序；等到处理完成，系统才会通过系统调用的方式，告诉应用程序 I/O 结果。

所谓异步 I/O，是指收到 I/O 请求后，系统会先告诉应用程序 I/O 请求已经收到，随后再去异步处理；等处理完成后，系统再通过事件通知的方式，告诉应用程序结果。

你可以看出，阻塞 / 非阻塞和同步 / 异步，其实就是两个不同角度的 I/O 划分方式。它们描述的对象也不同，阻塞 / 非阻塞针对的是 I/O 调用者（即应用程序），而同步 / 异步针对的是 I/O 执行者（即系统）。

我举个例子来进一步解释下。比如在 Linux I/O 调用中，

系统调用 `read` 是同步读，所以，在没有得到磁盘数据前，`read` 不会响应应用程序。

而 `aio_read` 是异步读，系统收到 AIO 读请求后不等处理就返回了，而具体的 `read` 结果，再通过回调异步通知应用程序。

再如，在网络套接字的接口中，

使用 `send()` 直接向套接字发送数据时，如果套接字没有设置 `O_NONBLOCK` 标识，那么 `send()` 操作就会一直阻塞，当前线程也没法去做其他事情。

当然，如果你用了 `epoll`，系统会告诉你这个套接字的状态，那就可以用非阻塞的方式使用。当这个套接字不可写的时候，你可以去做其他事情，比如读写其他套接字。

## 问题 2：“文件系统”课后思考



倪朋飞

摘录于 2019/01/27

最后，给你留一个思考题。在实际工作中，我们经常会根据文件名字，查找它所在路径，比如：

```
$ find / -name file-name
```

复制代码

今天的问题就是，这个命令，会不会导致系统的缓存升高呢？如果有影响，又会导致哪种类型的缓存升高呢？

引自：Linux性能优化实战

23 | 基础篇：Linux 文件系统是怎么工作的？

识别二维码打开原文  
「极客时间」App



在[文件系统原理文章](#)的最后，我给你留了一道思考题，那就是执行 find 命令时，会不会导致系统的缓存升高呢？如果会导致，升高的又是哪种类型的缓存呢？

关于这个问题，白华和 coyang 的答案已经很准确了。通过学习 Linux 文件系统的原理，我们知道，文件名以及文件之间的目录关系，都放在目录项缓存中。而这是一个基于内存的数据结构，会根据需要动态构建。所以，查找文件时，Linux 就会动态构建不在缓存中的目录项结构，导致 dentry 缓存升高。

## 白华

写于 2019/01/11

课后题：我找了一个目录下的文件，用的这个命令 `find / -type f -name copyright` 然后slabtop观察，发现dentry的SLABS和SIZE有了明显的提高，所以引起了目录项缓存的升高。在开始的时候dentry有一定的大小，我认为是缓存了/目录下系统基本的目录，但是系统后面下载、创建的内容是没有缓存的，使用查找命令会把这些都查找到然后缓存起来，所以使用find查找大量内容时候会造成性能下降。

前面看老男孩视频时候了解了inode和block。inode存储这些数据属性信息的，包含不限于文件大小、文件类型、文件权限、拥有者、硬链接数、所属组、修改时间，还包含指向文件实体的指针功能（inode节点---block的对应关系），但是inode惟独不包含文件名。文件名不在inode里，在上级目录的block里；Block来存储实际数据用的，例如照片、视频等普通文件数据。

今天看到了dentry，定义是用来记录文件的名字、索引节点指针以及与其他目录项的关联关系。但是和老男孩老师讲的有所区别，希望老师帮我解惑

引自：Linux性能优化实战

23 | 基础篇：Linux 文件系统是怎么工作的？



识别二维码打开原文  
「极客时间」App



## coyang

写于 2019/01/11

课后题：

这个命令，会不会导致系统的缓存升高呢？

--> 会的

如果有影响，又会导致哪种类型的缓存升高呢？

--> /xfs\_inode/ proc\_inode\_cache/dentry/inode\_cache

实验步骤：

1. 清空缓存：echo 3 > /proc/sys/vm/drop\_caches ; sync

2. 执行find：find / -name test

3. 发现更新top 4 项是：

OBJS	ACTIVE	USE	OBJ SIZE	SLABS	OBJ/S	LAB	CACHE	SIZE	NAME
37400	37400	100%	0.94K	2200	17	3	5200K	xfs_inode	
36588	36113	98%	0.64K	3049	12	24	392K	proc_inode_cache	
104979	104979	100%	0.19K	4999	21		19996K	dentry	
18057	18057	100%	0.58K	1389	13	1	1112K	inode_cache	

find / -name 这个命令是全盘扫描（既包括内存文件系统又包含本地的xfs【我的环境没有mount 网络文件系统】），所以 inode cache & dentry & proc inode cache 会升高。

另外，执行过了一次后再次执行find 就机会没有变化了，执行速度也快了很多，也就是下次的fin



d大部分是依赖cache的结果。


引自：Linux性能优化实战

23 | 基础篇：Linux 文件系统是怎么工作的？

识别二维码打开原文  
「极客时间」App



事实上，除了目录项缓存增加，Buffer 的使用也会增加。如果你用 vmstat 观察一下，会发现 Buffer 和 Cache 都在增长：

 复制代码

```
1 $ vmstat 1
2 procs -----memory----- ---swap-- -----io----- -system-- -----cpu-----
3  r  b    swpd    free    buff  cache    si    so    bi    bo    in    cs  us  sy  id  wa  st
4  0  1        0 7563744    6024 225944    0    0   3736    0   574 3249   3   5 89   3   0
5  1  0        0 7542792   14736 236856    0    0   8708    0 13494 32335   8 19 66   7   0
6  0  1        0 7494452   27280 272284    0    0  12544    0  4550 17084   5 15 68  13   0
7  0  1        0 7475084   42380 276320    0    0  15096    0  2541 14253   2   6 78  13   0
8  0  1        0 7455728   57600 280436    0    0  15220    0  2025 14518   2   6 70  22   0
```

这里，Buffer 的增长是因为，构建目录项缓存所需的元数据（比如文件名称、索引节点等），需要从文件系统中读取。

### 问题 3：“磁盘 I/O 延迟”课后思考

在[磁盘 I/O 延迟案例](#)的最后，我给你留了一道思考题。

我们通过 iostat，确认磁盘 I/O 已经出现了性能瓶颈，还用 pidstat 找出了大量磁盘 I/O 的进程。但是，随后使用 strace 跟踪这个进程，却找不到任何 write 系统调用。这是为什

么呢？



## 倪鹏飞

摘录于 2019/01/27

最后，给你留一个思考题，也是我在文章中提到过的，让你思考的问题。

今天的案例中，`iostat` 已经证明，磁盘 I/O 出现了性能瓶颈，`pidstat` 也证明了这个瓶颈是由 12280 号进程导致的。但是，`strace` 跟踪这个进程，却没有发现任何 `write` 系统调用。

这究竟是怎么回事？难道是因为案例使用的编程语言 Python 本身是解释型？还是说，因为案例运行在 Docker 中呢？

这里我小小提示一下。当你发现性能工具的输出无法解释时，最好返回去想想，是不是分析中漏掉了什么线索，或者去翻翻工具手册，看看是不是某些默认选项导致的。

引自：Linux性能优化实战

27 | 案例篇：为什么我的磁盘I/O延迟很高？

识别二维码打开原文  
「极客时间」App



很多同学的留言都准确回答了这个问题。比如，划时代和 jeff 的留言都指出，在这个场景中，我们需要加 -f 选项，以便跟踪多进程和多线程的系统调用情况。

# 划时代

写于 2019/01/21

赞同在strace -p PID后加上-f，多进程和多线程都可以跟踪。

引自：Linux性能优化实战

271 案例篇：为什么我的磁盘I/O延迟很高？

识别二维码打开原文  
「极客时间」App



jeff

写于 2019/01/21

写文件是由子线程执行的，所以直接strace跟踪进程没有看到write系统调用，可以通过pstree查看进程的线程信息，再用strace跟踪。或者，通过strace -fp pid 跟踪所有线程。

引自：Linux性能优化实战

271 案例篇：为什么我的磁盘I/O延迟很高？

识别二维码打开原文  
「极客时间」App



你看，仅仅是不恰当选项，都可能会导致性能工具“犯错”，呈现这种看起来不合逻辑的结果。非常高兴看到，这么多同学已经掌握了性能工具使用的核心思路——弄清楚工具本身的原理和问题。

## 问题 4：“MySQL 案例”课后思考

在 [MySQL 案例](#) 的最后，我给你留了一个思考题。

为什么 DataService 应用停止后，即使仍没有索引，MySQL 的查询速度还是快了很多，并且磁盘 I/O 瓶颈也消失了呢？



ninuxer	
写于 2019/01/23	打卡day29 echo 1>/proc/sys/vm/drop_caches表示释放pagecache，也就是文件缓存，而mysql读书的数据就是文件缓存，dataservice不停的释放文件缓存，就导致MySQL都无法利用磁盘缓存，也就慢了~
<div>引自：Linux性能优化实战</div> <div>28   案例篇：一个SQL查询要15秒，这是怎么回事？</div>	
<div>识别二维码打开原文 「极客时间」App</div> 	

ninuxer 的留言基本解释了这个问题，不过还不够完善。

事实上，当你看到 DataService 在修改 `/proc/sys/vm/drop_caches` 时，就应该想到前面学过的 Cache 的作用。

我们知道，案例应用访问的数据表，基于 MyISAM 引擎，而 MyISAM 的一个特点，就是只在内存中缓存索引，并不缓存数据。所以，在查询语句无法使用索引时，就需要数据表从数据库文件读入内存，然后再进行处理。

所以，如果你用 `vmstat` 工具，观察缓存和 I/O 的变化趋势，就会发现下面这样的结果：

```

1 $ vmstat 1
2
3 procs -----memory----- ---swap-- -----io----- -system-- -----cpu-----
4  r  b   swpd   free   buff  cache   si   so    bi   bo   in   cs us sy id wa st
5
6 # 备注: DataService 正在运行
7 0  1       0 7293416    132 366704    0    0 32516    12   36  546  1  3 49 48  0
8 0  1       0 7260772    132 399256    0    0 32640    0   37  463  1  1 49 48  0
9 0  1       0 7228088    132 432088    0    0 32640    0   30  477  0  1 49 49  0
10 0  0       0 7306560    132 353084    0    0 20572    4   90  574  1  4 69 27  0
11 0  2       0 7282300    132 368536    0    0 15468    0   32  304  0  0 79 20  0
12
13 # 备注: DataService 从这里开始停止
14 0  0       0 7241852    1360 424164    0    0   864   320  133 1266  1  1 94  5  0
15 0  1       0 7228956    1368 437400    0    0 13328    0   45  366  0  0 83 17  0
16 0  1       0 7196320    1368 470148    0    0 32640    0   33  413  1  1 50 49  0
17 ...
18 0  0       0 6747540    1368 918576    0    0 29056    0   42  568  0  0 56 44  0
19 0  0       0 6747540    1368 918576    0    0     0     0   40  141  1  0 100  0  0

```

在 DataService 停止前，cache 会连续增长三次后再降回去，这正是因为 DataService 每隔 3 秒清理一次页缓存。而 DataService 停止后，cache 就会不停地增长，直到增长为 918576 后，就不再变了。

这时，磁盘的读（bi）降低到 0，同时，iowait（wa）也降低到 0，这说明，此时的所有数据都已经在系统的缓存中了。我们知道，缓存是内存的一部分，它的访问速度比磁盘快得多，这也能解释，为什么 MySQL 的查询速度变快了很多。

从这个案例，你会发现，MySQL 的 MyISAM 引擎，本身并不缓存数据，而要依赖系统缓存来加速磁盘 I/O 的访问。一旦系统中还有其他应用同时运行，MyISAM 引擎就很难充分利用系统缓存。因为系统缓存可能被其他应用程序占用，甚至直接被清理掉。

所以，一般来说，我并不建议，把应用程序的性能优化完全建立在系统缓存上。还是那句话，最好能在应用程序的内部分配内存，构建完全自主控制的缓存，比如 MySQL 的 InnoDB 引擎，就同时缓存了索引和数据；或者，可以使用第三方的缓存应用，比如 Memcached、Redis 等。

今天主要回答这些问题，同时也欢迎你继续在留言区写下疑问和感想，我会持续不断地解答。希望借助每一次的答疑，可以和你一起，把文章知识内化为你的能力，我们不仅在实战

中演练，也要在交流中进步。



# Linux 性能优化实战

10 分钟帮你找到系统瓶颈

倪朋飞

微软资深工程师  
Kubernetes 项目维护者



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 31 | 套路篇：磁盘 I/O 性能优化的几个思路

下一篇 加餐（一） | 书单推荐：性能优化和Linux 系统原理

## 精选留言 (14)

写留言



ninuxer

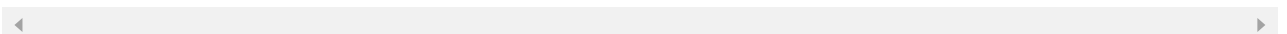
2019-02-01

5

打卡day33

感恩作者带来的分享，提前祝新年快乐！

作者回复：新年快乐！



eagle

5



2019-03-26

4

我根据我们自己实际应用中遇到的情况，试着回复一下两个问题：

安小依 的问题，df -h 显示占用100%，而关闭应用程序后，再次df -h是85%，这一般是因为该应用程序还有指向已删除文件的文件指针没有关闭，典型的比如日志文件，虽然在操作系统中用rm命令删除了，在相应的目录中已经没有该文件了，但如果应用中还有对应的文件指针没有关闭，则实际硬盘空间还不会释放，而应用程序被关闭时，实际空间才...  
展开 ▾

作者回复: 谢谢 😊



Maxwell

2019-03-14

👍 1

Windows和linux有很大区别吧？如果想深入了解windows，有什么可以推荐的书吗？

作者回复: Windows书籍最推荐的是《Windows Internals 7th edition》



安小依

2019-03-01

👍 1

老师，今天遇见了一个问题: 系统使用 df -h 显示磁盘占用100%了，而且应用程序(这是一个不停下载 apk 文件、解压缩并分析 apk文件的应用程序)在命令行也提示磁盘空间不足了。但是，关闭应用程序后，再次 df-h 统计，却发现这次磁盘占用是 85%，释放了 15% 大约150G 的空间...能大概推测出来为什么关闭应用后，磁盘空间突然多了的原因吗？

展开 ▾

作者回复: 解压缩很可疑，有没有看看这些apk解压后的大小？



Ivy

2019-02-19

👍 1

老师您好，我最近在生产环境遇到一个问题，centos7频繁报错tcp out of memory，访问页面时css文件响应头200，但是响应正文为空，我猜测就是因为tcp问题，有时候又能正常返回，每次重启php fpm就能解决问题，cat /proc/net/sockstat 的时候tcp 行mem值在fpm重启前后差距很大，同时tw状态的连接也很多，alloc也很大，我该怎么去找原因？

能看到每个tw状态的连接占用多少tcp 内存吗？或者怎么查询php fpm为何没有释放tcp...

展开 ∨

作者回复: 可以考虑调整 tcp\_max\_tw\_buckets、ip\_contrack\_max、ip\_contrack 这些内核选项



Leon 📷

2019-02-01

👍 1

老师，读文件系统的内容不会引起buffer升高吧，读块设备会引起，我做了文章的实验发现

```
r b swpd free buff cache si so bi bo in cs us sy id wa st
1 0 0 1620788 0 431512 0 0 348 70 415 585 2 4 94 0 0
0 0 0 1618488 0 431948 0 0 480 0 1605 2056 1 4 96 0 0...
```

展开 ∨

作者回复: 是的，读文件内容的时候不会的。文中指的是执行 find 命令查找文件的情景



Stephen

2019-04-03

👍

《UNIX网络编程》提到的5种IO模型中，除了异步IO模型没有阻塞操作外，其他四种IO模型（阻塞IO、非阻塞IO、IO多路复用、信号驱动IO）都有阻塞操作。是不是可以这么理解：同步IO一定有阻塞可能有非阻塞，异步IO一定是非阻塞；有阻塞一定是同步IO，...

展开 ∨



allan

2019-03-24

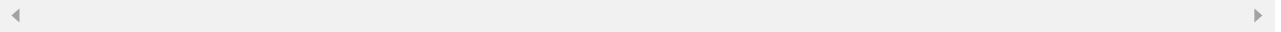
👍

原文：DataService 停止后，bi iowait 都降到0，说明此时的所有数据都已经在系统的缓存中了。

这里所有数据指的是 数据库文件 中的数据 是吗？不包括索引。

展开 ∨

作者回复: 嗯，数据库文件



如果

2019-03-20



DAY32，打卡

展开 ▾



马殿军

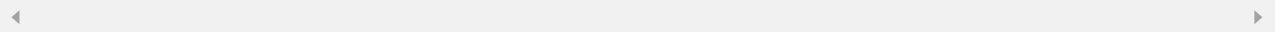
2019-02-20



老师好，请教：目录项缓存在cache中，索引节点缓存在buffer中，这是对的吗？

还是二者都在buffer中？

作者回复: 不是，buffer只是磁盘的缓存，其他都可以认为是cache



陈帅

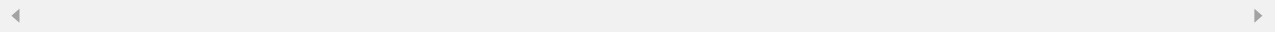
2019-02-11



关于，阻塞、非阻塞 I/O 与同步、异步 I/O，这个问题回答。我确认下，是不是其实是一个东西，只不过划分的角度不同罢了。是这个意思吗？

展开 ▾

作者回复: 是的，在很多场景下都可以理解成一样的。不过要细分的话，还是有些区别（见文中内容）



我来也

2019-02-02




[D32打卡]

时间过得真快，转眼专栏的五大模块已经学完了三个。 😊

展开 ▾



作者回复: 



Leon 

2019-02-01



老师，我们的测试环境机器我从几个指标看只有系统盘每秒写的数据量比测试环境多，为什么比测试环境卡很多，进程也只是测试环境一倍而已，使用vmstat pidstat,top,发现只有线上机器进程数多一倍，io写入量是测试机器10倍，测试配置4核16G，线上32核，256G，磁盘随机读写都是79MB/s左右

测试17时50分54秒 0 1 3.85 16.61 4.86 systemd...

展开 

作者回复: 除了CPU、内存、磁盘之外，网络也可能是个原因。另外，对 I/O，还可以用 iostat 看一下其他指标是不是有什么线索



blackpikle...

2019-02-01



老师，在工作中遇到了 Ubuntu 16.04 系统死机的问题，和性能优化并不直接相关，不过还是想问一下遇到这种问题该如何分析。我能想到的步骤是：

1. 看 /var/crash 下是否有 kernel panic 的记录；
2. 看 /var/log/syslog 下是否有应用程序异常记录；
3. 看服务器上主要的应用程序日志，是否有异常；...

展开 

作者回复: 系统日志肯定是最有效的，一般都会留下一些线索。死机的时候，如果可以通过IPMI看到服务器的Console，那也有可能看到一些线索（比如内核中的错误或者内核栈等等）。