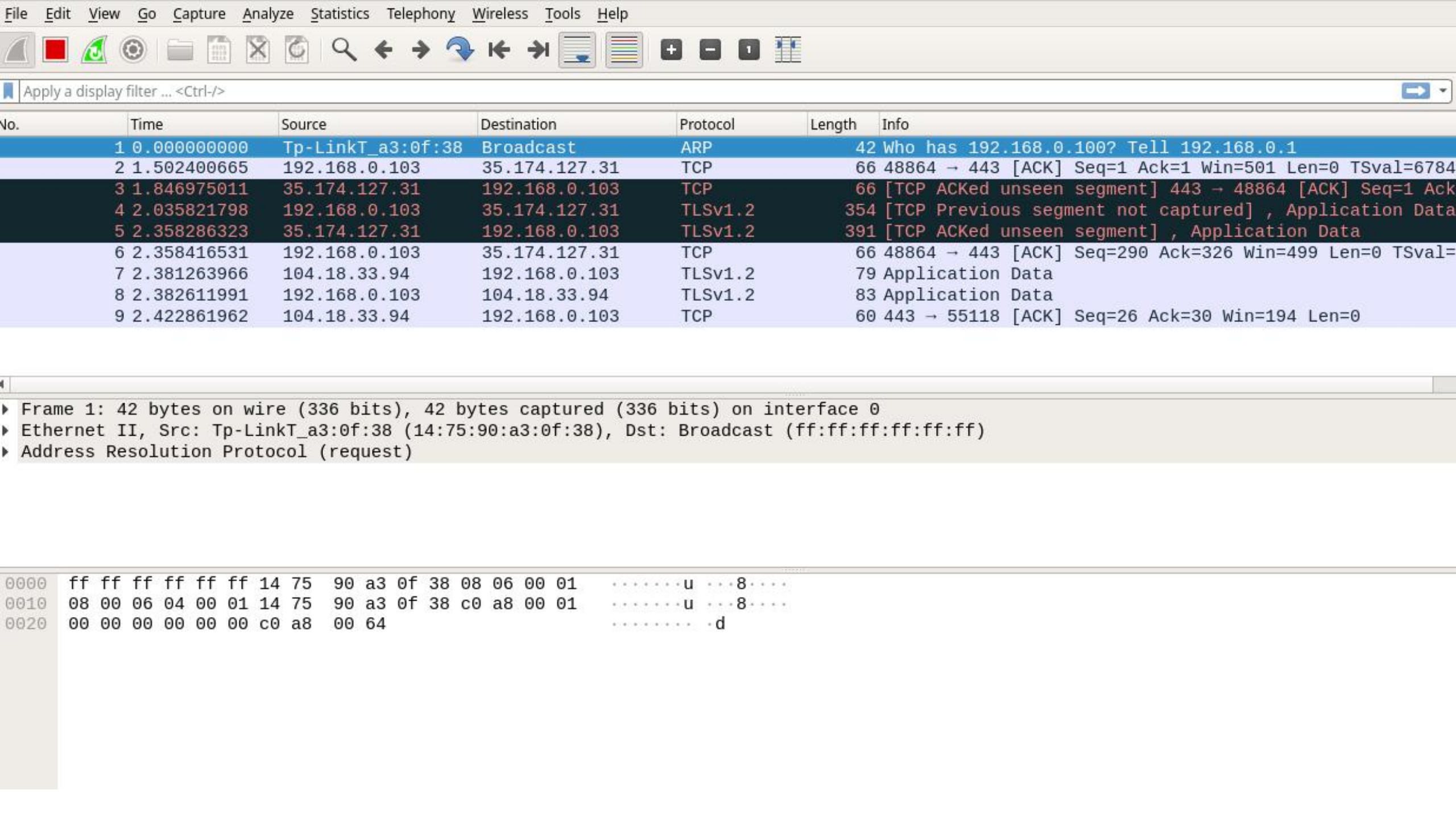
一段沉浸式解决Linux性能问题的经历

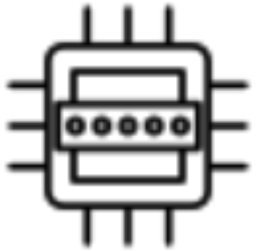
```
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot# reboot
```

```
top - 10:49:03 up 26 days,  2:19,  1 user,  load average: 0.56, 0.31, 0.21
Tasks: 380 total,   1 running, 304 sleeping,   0 stopped,   1 zombie
%Cpu(s):  2.8 us,  2.3 sy,  0.0 ni, 94.2 id,  0.4 wa,  0.0 hi,  0.3 si,  0.0 st
KiB Mem :  7924112 total,   343072 free,  2722372 used,  4858668 buff/cache
KiB Swap:  2097148 total,   562460 free,  1534688 used.  4563840 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 2103 root      20   0  470928  51140  19024 S   9.6  0.6 485:58.40 Xorg
 1548 root      20   0  825400   6960   5208 S   6.0  0.1   2376:57 sunloginc+
19339 jinxin    20   0  368024  17316  15032 S   2.6  0.2   0:00.17 xfce4-scr+
 5433 jinxin    20   0  448320  29676  13388 S   1.3  0.4  65:17.56 xfwm4
 2725 root      20   0 2632928  19980   3916 S   1.0  0.3 437:23.32 taosd
19320 root      20   0   47868   4472   3660 R   1.0  0.1   0:00.15 top
  346 root      20   0   48068   4064   2408 S   0.7  0.1  82:12.20 systemd-u+
 8065 jinxin    20   0 16.970g 246528 118772 S   0.7  3.1 878:12.51 chrome
30201 root      20   0 1666732 139516 110792 S   0.7  1.8   1:17.04 mongod
    1 root      20   0  225972   7804   5056 S   0.3  0.1  34:40.54 systemd
    8 root      20   0       0      0      0 I   0.3  0.0  78:05.80 rcu_sched
  320 root      19  -1  264732 126996 112060 S   0.3  1.6  16:20.19 systemd-j+
 1085 message+  20   0   51716   5316   3348 S   0.3  0.1  89:55.71 dbus-daem+
```

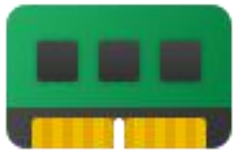
```
    1 root      20   0  225972   7804   5056 S   0.3   0.1  34:40.55 systemd
    8 root      20   0       0      0      0 I   0.3   0.0  78:05.82 rcu_sched
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot# free
              total        used        free      shared  buff/cache   available
Mem:        7924112     2718040      380300      305916     4825772     4601672
Swap:       2097148     1534688      562460
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
root@jinxin:/tftpboot#
```


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 Tp-LinkT_a3:0f:38 Broadcast ARP 42 Who has 192.168.0.100? Tell 192.168.0.1
2 1.502400665 192.168.0.103 35.174.127.31 TCP 66 48864 → 443 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=6784
3 1.846975011 35.174.127.31 192.168.0.103 TCP 66 [TCP ACKed unseen segment] 443 → 48864 [ACK] Seq=1 Ack
4 2.035821798 192.168.0.103 35.174.127.31 TLSv1.2 354 [TCP Previous segment not captured] , Application Data
5 2.358286323 35.174.127.31 192.168.0.103 TLSv1.2 391 [TCP ACKed unseen segment] , Application Data
6 2.358416531 192.168.0.103 35.174.127.31 TCP 66 48864 → 443 [ACK] Seq=290 Ack=326 Win=499 Len=0 TSval=
7 2.381263966 104.18.33.94 192.168.0.103 TLSv1.2 79 Application Data
8 2.382611991 192.168.0.103 104.18.33.94 TLSv1.2 83 Application Data
9 2.422861962 104.18.33.94 192.168.0.103 TCP 60 443 → 55118 [ACK] Seq=26 Ack=30 Win=194 Len=0
Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
Ethernet II, Src: Tp-LinkT_a3:0f:38 (14:75:90:a3:0f:38), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
0000 ff ff ff ff ff ff 14 75 90 a3 0f 38 08 06 00 01
0010 08 00 06 04 00 01 14 75 90 a3 0f 38 c0 a8 00 01
0020 00 00 00 00 00 00 c0 a8 00 64



CPU

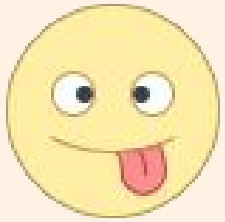
I/O



内存

网络

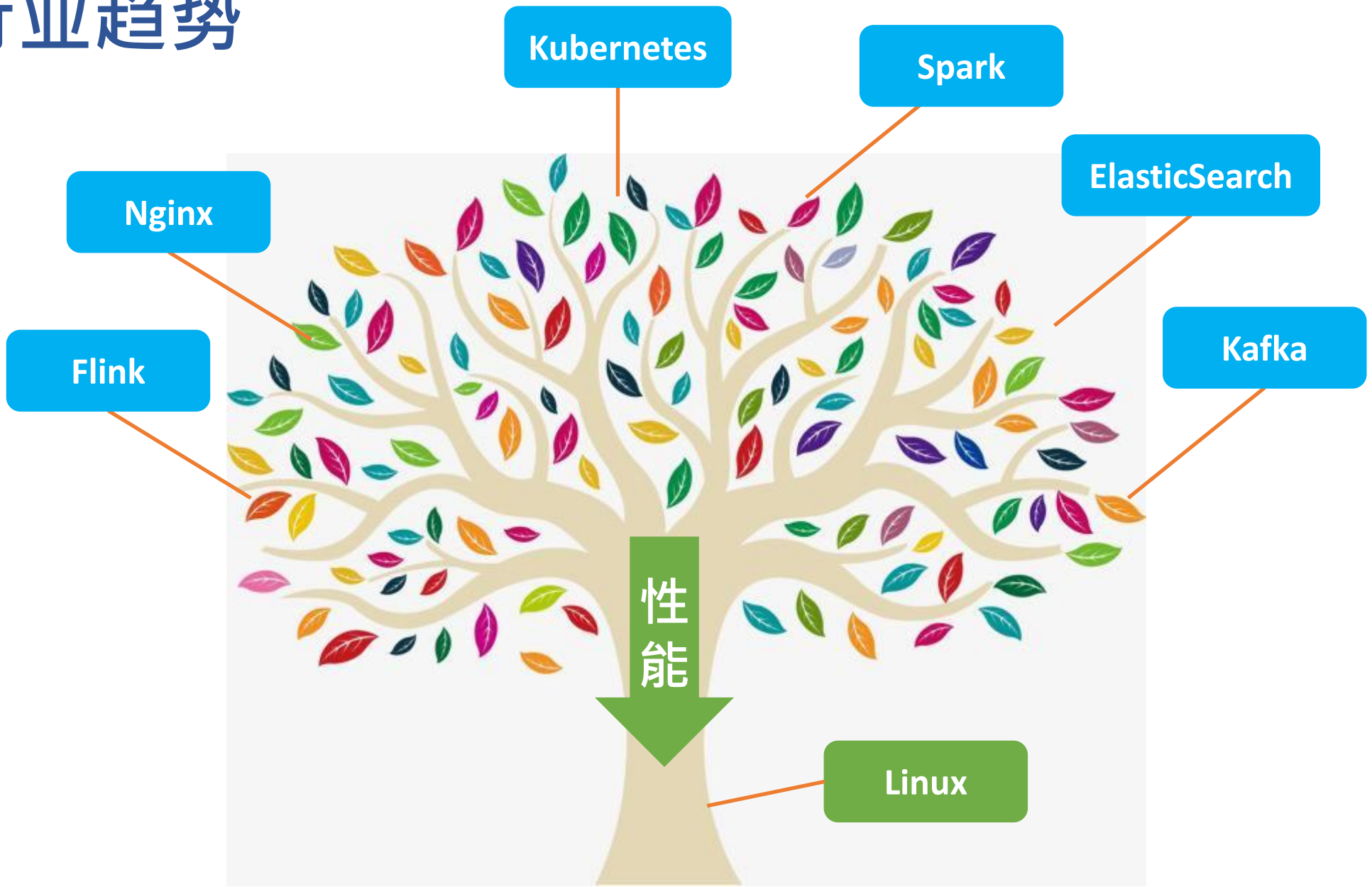
事关linux性能，你的抓狂时刻 ...



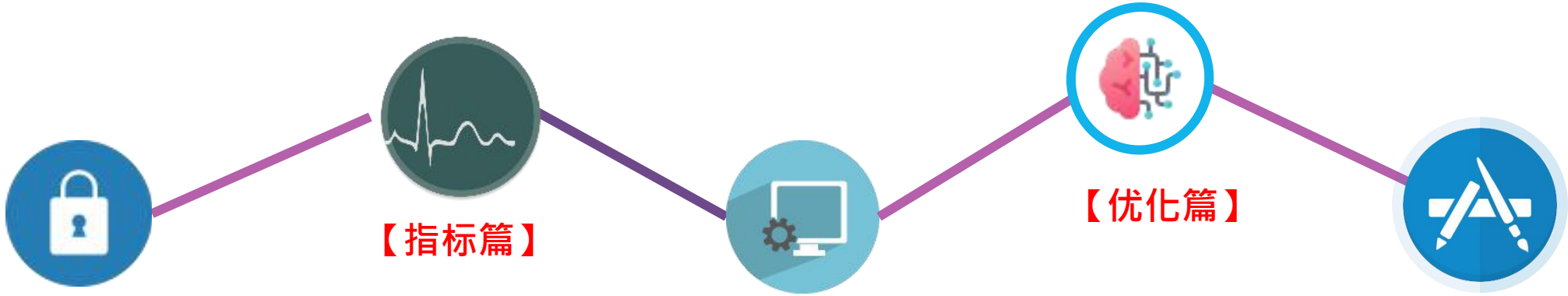
- 程序开发完毕，满足了功能性要求，但是性能指标没达到，该如何优化？【开发】
- 针对开发人员开发的程序，如何通过一些工具进行有效的测试，发现其中的性能瓶颈？【测试】
- 容器环境下，有些调试工具不全或者压根没有，该怎么进行调试？【运维】
- 随着时间的推移，线上的核心服务反应变慢，在服务不能停止的情况下如何进行分析和调试？【运维】
- 找到了性能瓶颈，但是不了解 linux 底层的运作机制，导致不知道该如何入手？【开发】

行业趋势

- 云平台
- 大数据
- 机器学习
- 边缘智能



课程内容：4大模块，5个维度 68+作品



思维导图，联想记忆

1 张脑图，囊括所有，随时参考

45+ linux下的性能工具用法介绍

40+ 性能调优建议

10+ 案例，一起实践调优之法

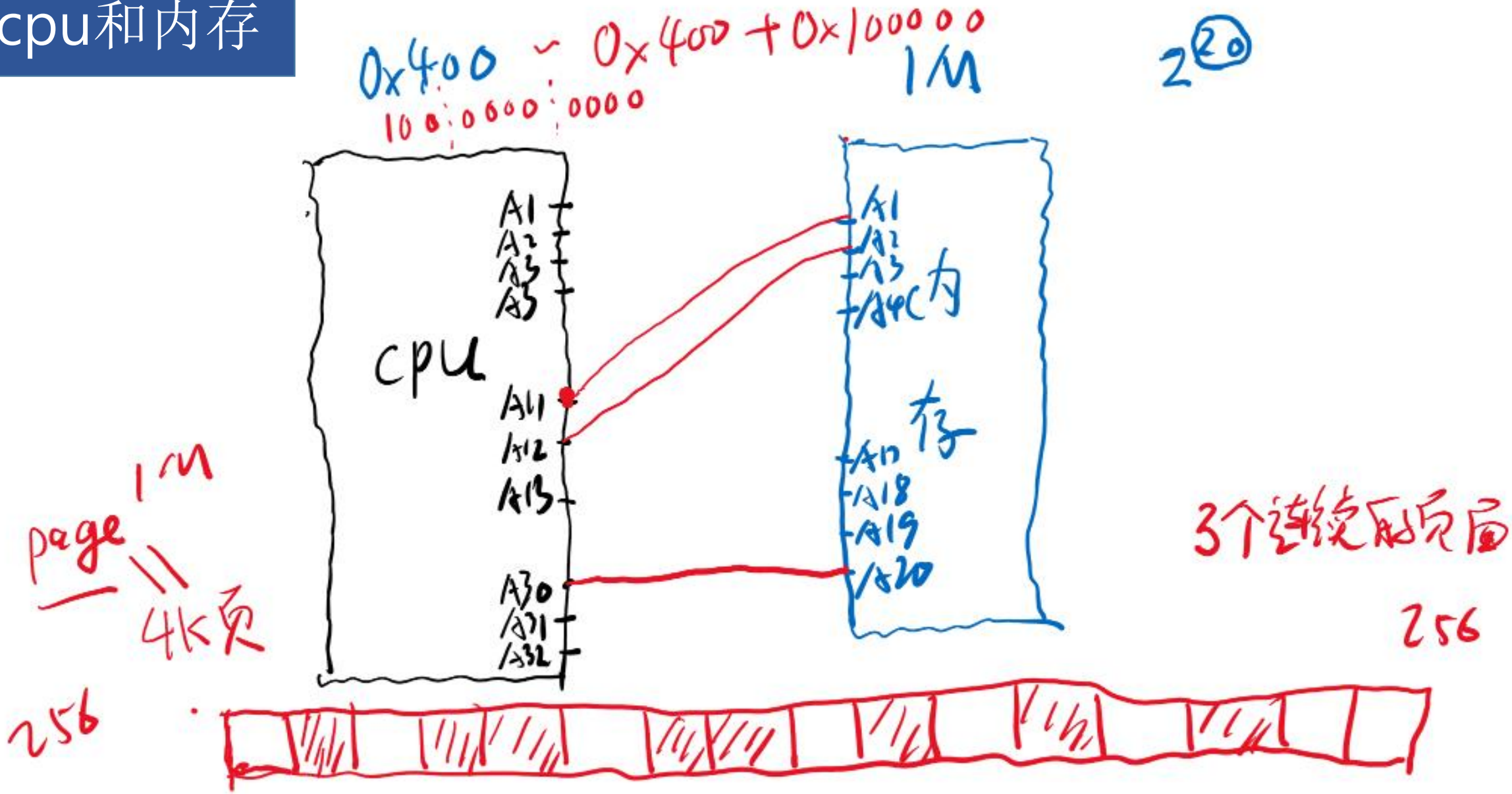
课程目标

5分钟内解决90%的性能问题！

课程适合人群

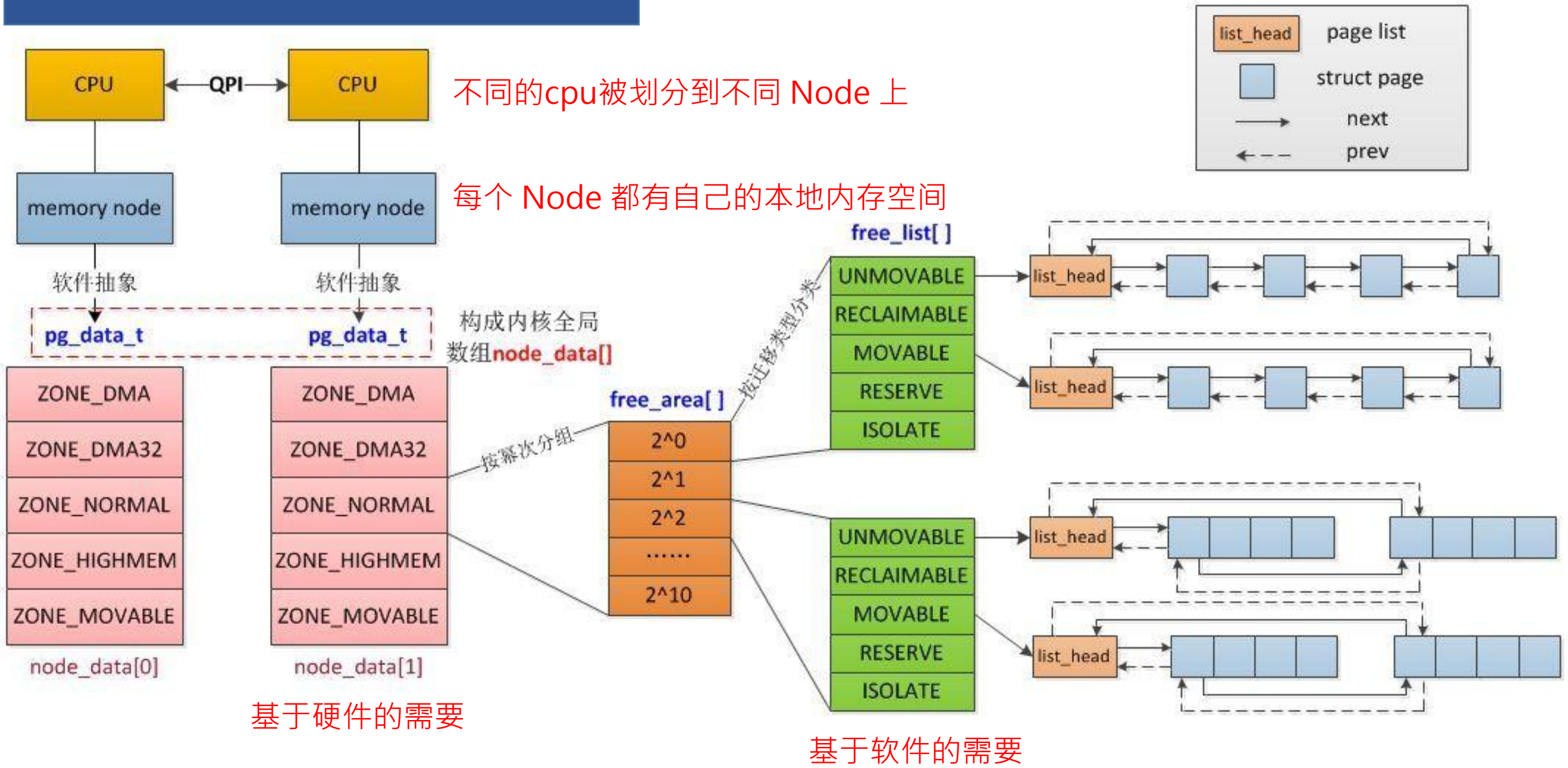
- **运维人员**，在拿不到源代码的情况下，定位线上应用的性能问题；
- 一般**开发人员**，需要在编码阶段就采用高性能的做法，防患于未然；
- **测试人员**，通过一些性能指标，评估软件产品的改进程度；
- **大数据开发者**，想从纷繁复杂的大数据系统中，找到影响性能的木桶短板，从而达到四两拨千斤的效果；
- **容器云开发者**，想知道一些针对容器环境的性能调优方法和套路；

cpu和内存



Flat Model

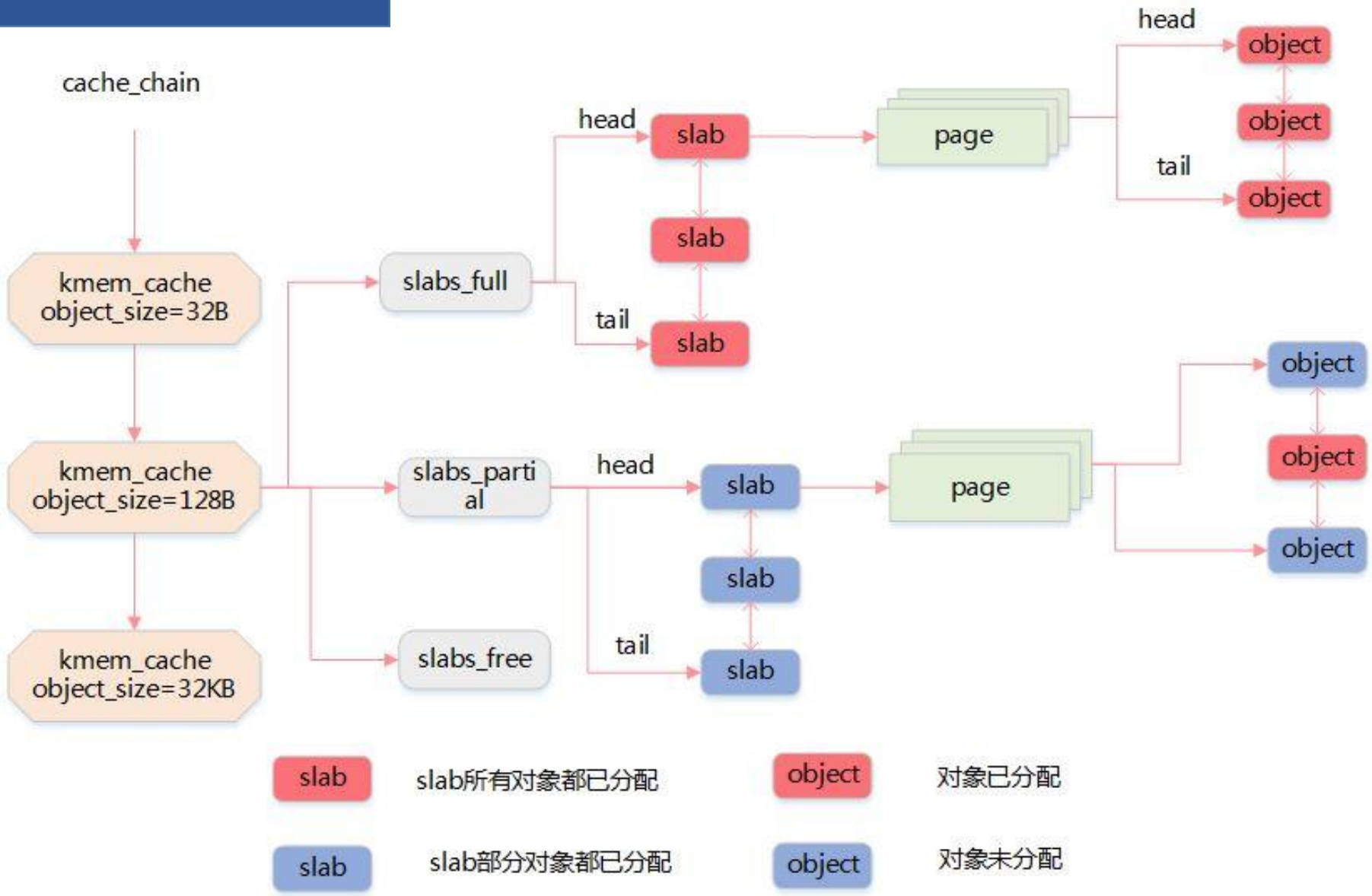
页框和伙伴算法



linux内核内存管理机制

宏	类型
MIGRATE_UNMOVABLE	不可移动页
MIGRATE_MOVABLE	可移动页
MIGRATE_RECLAIMABLE	可回收页
MIGRATE_PCPTYPES	是per_cpu_pageset, 即用来表示每CPU页框高速缓存的数据结构中的链表的迁移类型数目
MIGRATE_HIGHATOMIC	= MIGRATE_PCPTYPES, 在罕见的情况下，内核需要分配一个高阶的页面块而不能休眠.如果向具有特定可移动性的列表请求分配内存失败，这种紧急情况下可从MIGRATE_HIGHATOMIC中分配内存
MIGRATE_CMA	Linux内核最新的连续内存分配器(CMA), 用于避免预留大块内存
MIGRATE_ISOLATE	是一个特殊的虚拟区域, 用于跨越NUMA结点移动物理内存页. 在大型系统上, 它有益于将物理内存页移动到接近于使用该页最频繁的CPU.
MIGRATE_TYPES	只是表示迁移类型的数目, 也不代表具体的区域

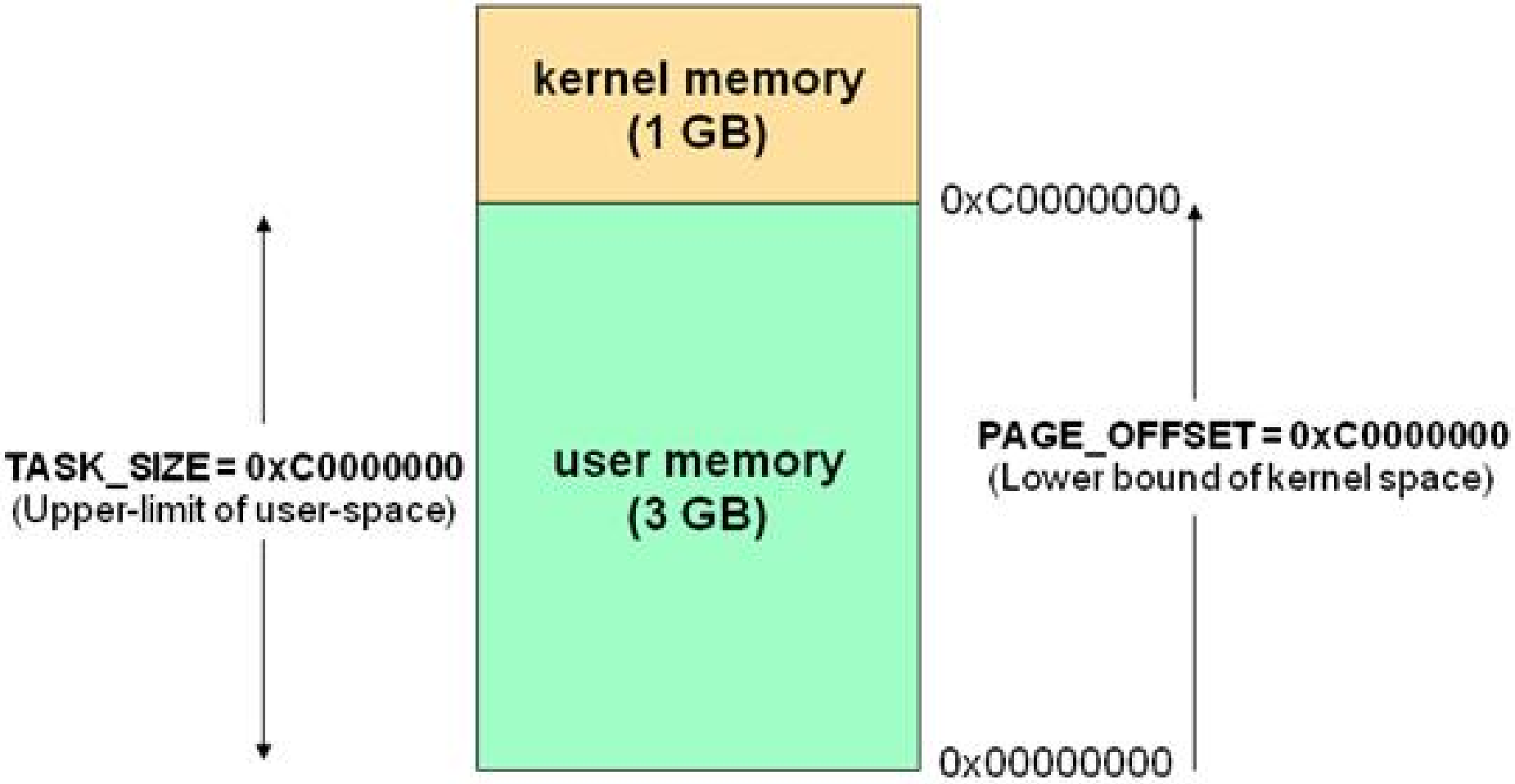
通过slab管理小内存



linux内核内存管理机制

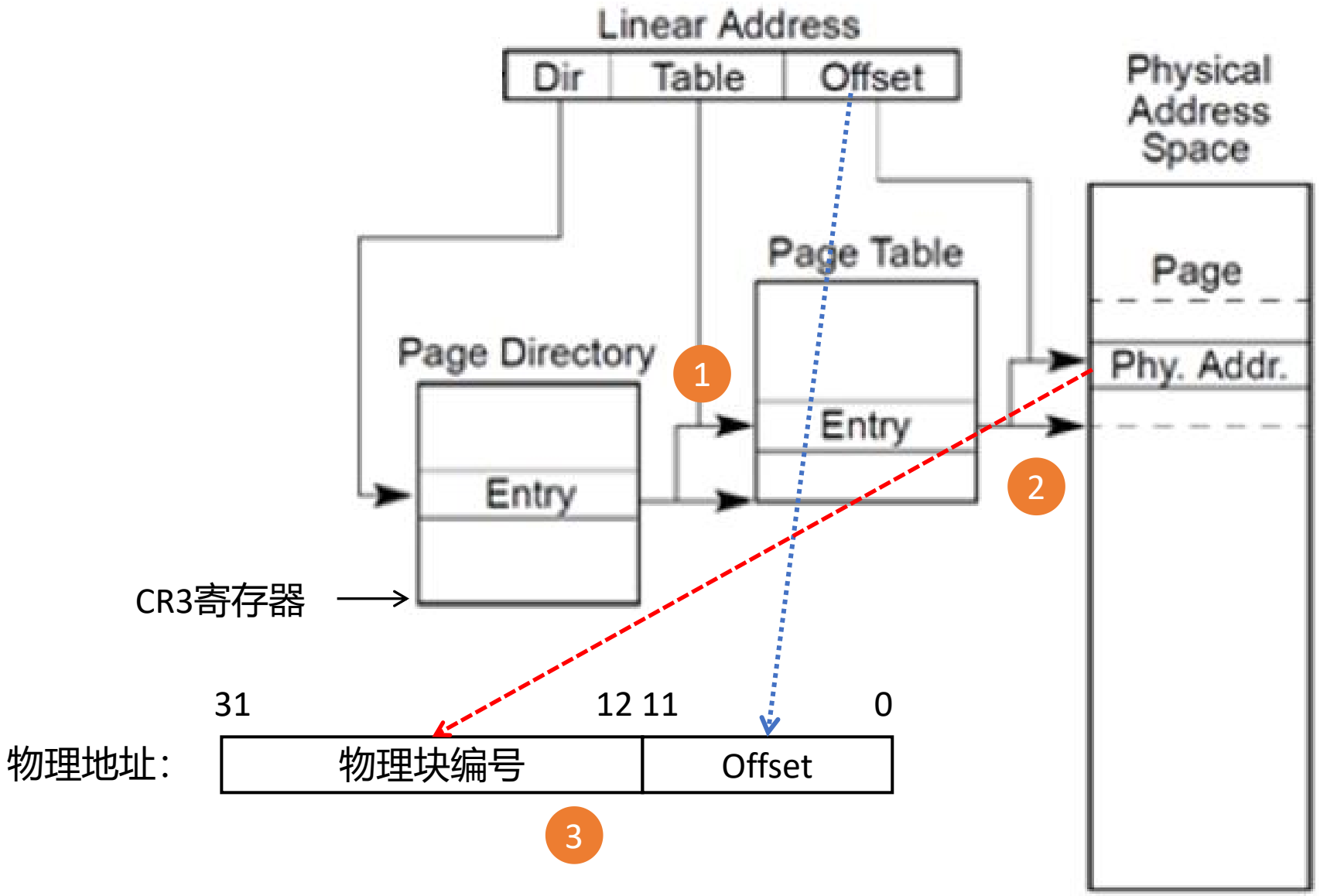
linux内核代码能不能直接访问物理内存地址？

linux虚拟地址空间机制



32系统内核空间和用户空间地址范围

linux二级页表映射原理



线性地址：0xC1234567

高10：0x304
中10：0x234
低12：0x567

0x123左移12位+0x567
物理地址：0x00123567

linux虚拟地址空间机制

进程的3G的虚拟地址空间只有映射为物理地址空间，才能够被使用！

进程如何管理和分配它的3G的虚拟地址空间呢？

分治思想

linux虚拟地址空间机制

按照不同的访问属性和功能划分为不同的内存区域，我们也叫虚拟内存区域（VMA）

hello world

- 代码段：可执行文件的内存映射
- 数据段：可执行文件的已初始化全局变量和静态局部变量的内存映射；
- bss段：未初始化的或者值为0的变量的内存映射；
- lib库的代码段；（多个）
- lib库的数据段；（多个）
- lib库的bss段；（多个）
- 任何内存映射文件（有名mmap建立）；
- 任何共享内存段（匿名mmap建立）；
- 进程栈**stack**；
- 进程堆**heap**；

```c
#include <stdio.h>
#include <unistd.h>>
void main()
{
        char *buffer = NULL;
        buffer = (char *)malloc(2 * sizeof(char));
        printf( "PID = %d\n", getpid());
        while(1) {
                sleep(2);
        }
```

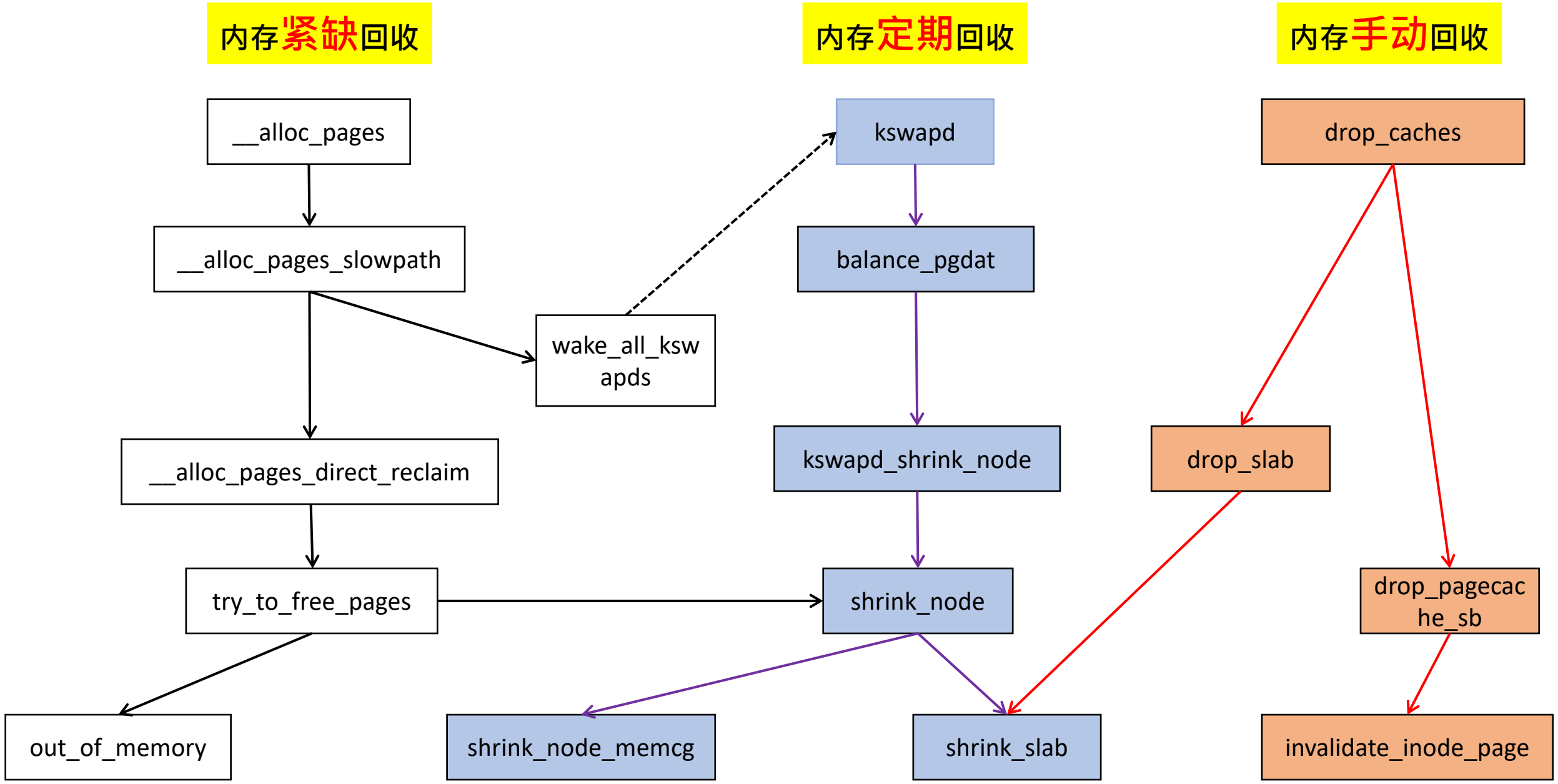
```
root@ubuntu:/home/jinxin/linux-4.9.229# cat /proc/11356/maps
00400000-00401000 r-xp 00000000 08:01 938729                             /home/jinxin/app/main
00600000-00601000 r--p 00000000 08:01 938729                             /home/jinxin/app/main
00601000-00602000 rw-p 00001000 08:01 938729                             /home/jinxin/app/main
01cdd000-01cfe000 rw-p 00000000 00:00 0                                  [heap]
7fcc1261e000-7fcc127dc000 r-xp 00000000 08:01 1462783                    /lib/x86_64-linux-gnu/libc-2.19.so
7fcc127dc000-7fcc129dc000 ---p 001be000 08:01 1462783                    /lib/x86_64-linux-gnu/libc-2.19.so
7fcc129dc000-7fcc129e0000 r--p 001be000 08:01 1462783                    /lib/x86_64-linux-gnu/libc-2.19.so
7fcc129e0000-7fcc129e2000 rw-p 001c2000 08:01 1462783                    /lib/x86_64-linux-gnu/libc-2.19.so
7fcc129e2000-7fcc129e7000 rw-p 00000000 00:00 0
7fcc129e7000-7fcc12a0a000 r-xp 00000000 08:01 1462780                    /lib/x86_64-linux-gnu/ld-2.19.so
7fcc12bee000-7fcc12bf1000 rw-p 00000000 00:00 0
7fcc12c08000-7fcc12c09000 rw-p 00000000 00:00 0
7fcc12c09000-7fcc12c0a000 r--p 00022000 08:01 1462780                    /lib/x86_64-linux-gnu/ld-2.19.so
7fcc12c0a000-7fcc12c0b000 rw-p 00023000 08:01 1462780                    /lib/x86_64-linux-gnu/ld-2.19.so
7fcc12c0b000-7fcc12c0c000 rw-p 00000000 00:00 0
7ffdf8474000-7ffdf8495000 rw-p 00000000 00:00 0                          [stack]
7ffdf854e000-7ffdf8550000 r--p 00000000 00:00 0                          [vvar]
7ffdf8550000-7ffdf8552000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

内核每进程的vm_area_struct项	/proc/pid/maps中的项	含义
vm_start	“-”前一列，如00377000	此段虚拟地址空间起始地址
vm_end	“-”后一列，如00390000	此段虚拟地址空间结束地址
vm_flags	第三列，如r-xp	此段虚拟地址空间的属性。每种属性用一个字段表示，r表示可读，w表示可写，x表示可执行，p和s共用一个字段，互斥关系，p表示私有段，s表示共享段，如果没有相应权限，则用'-'代替
vm_pgoff	第四列，如00000000	对有名映射，表示此段虚拟内存起始地址在文件中以页为单位的偏移。对匿名映射，它等于0或者vm_start/PAGE_SIZE
vm_file->f_dentry->d_inode->i_sb->s_dev	第五列，如fd:00	映射文件所属设备号。对匿名映射来说，因为没有文件在磁盘上，所以没有设备号，始终为00:00。对有名映射来说，是映射的文件所在设备的设备号
vm_file->f_dentry->d_inode->i_ino	第六列，如9176473	映射文件所属节点号。对匿名映射来说，因为没有文件在磁盘上，所以没有节点号，始终为00:00。对有名映射来说，是映射的文件的节点号
	第七列，如/lib/ld-2.5.so	对有名来说，是映射的文件名。对匿名映射来说，是此段虚拟内存在进程中的角色。[stack]表示在进程中作为栈使用，[heap]表示堆。其余情况则无显示

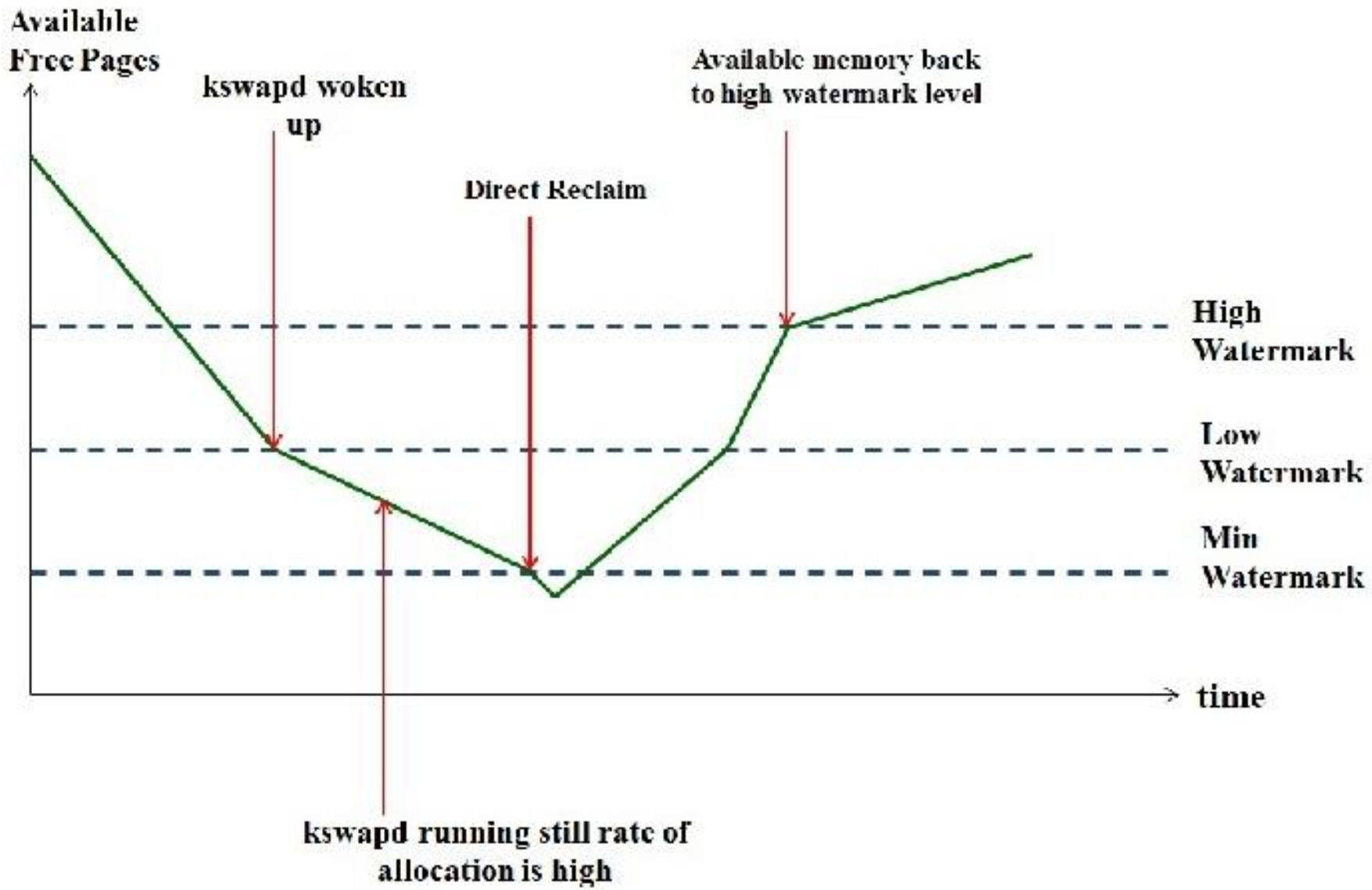
进程使用的内存分类

	Private	Shared
	1	2
Anonymous	. stack . malloc() . brk()/sbrk() . mmap(PRIVATE, ANON)	. POSIX shm* . mmap(SHARED, ANON)
File-backed	. mmap(PRIVATE, fd) . pgms/shared libs	. mmap(SHARED, fd)
	3	4

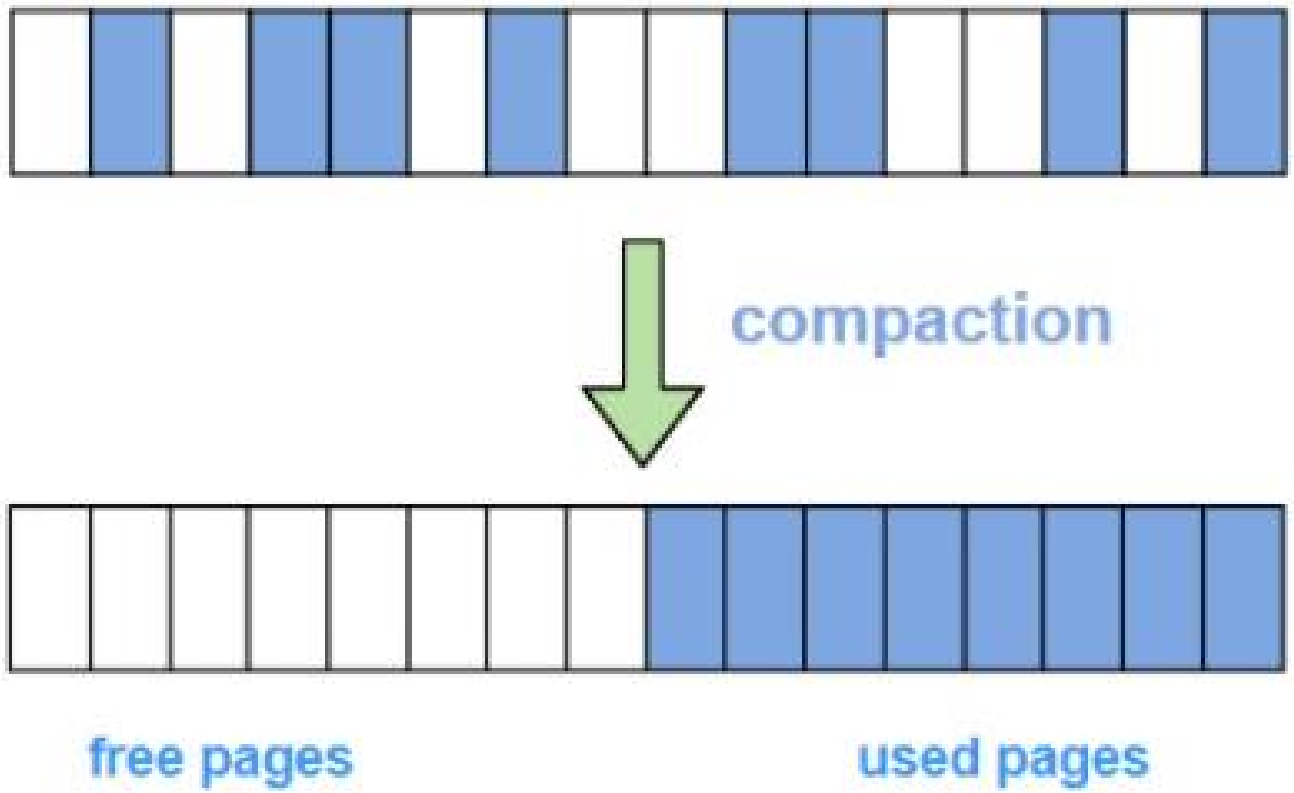
4.9.229 based

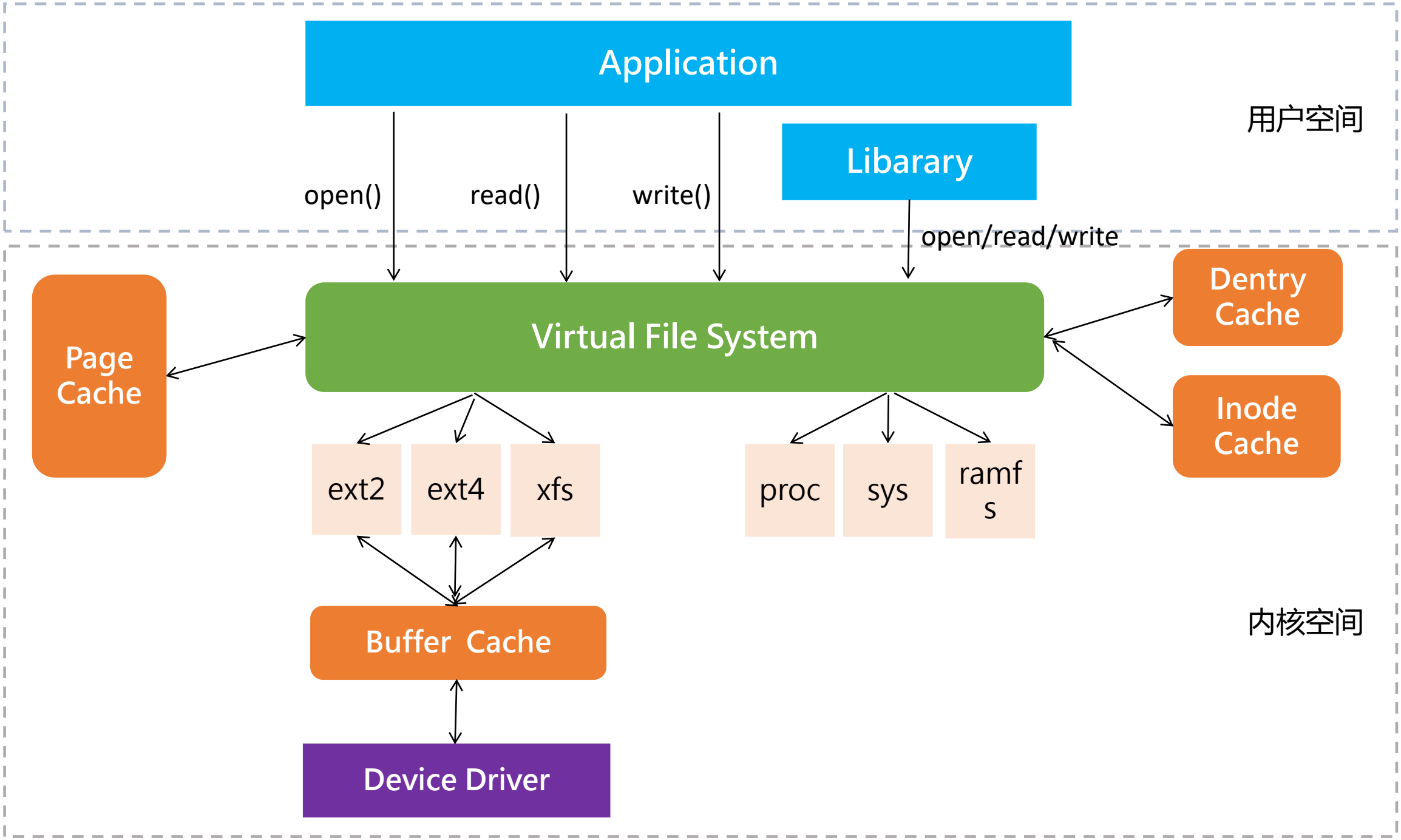

内存紧缺回收
内存定期回收
内存手动回收
__alloc_pages
__alloc_pages_slowpath
__alloc_pages_direct_reclaim
try_to_free_pages
out_of_memory
wake_all_kswapds
kswapd
balance_pgdat
kswapd_shrink_node
shrink_node
shrink_node_memcg
shrink_slab
drop_caches
drop_slab
drop_pagecache_sb
invalidate_inode_page

Zone Watermark



Memory Migrate




Application
用户空间
Libarary
open()
read()
write()
open/read/write
Page Cache
Virtual File System
Dentry Cache
Inode Cache
ext2
ext4
xfs
proc
sys
ramfs
Buffer Cache
Device Driver
内核空间

文件系统结构概览

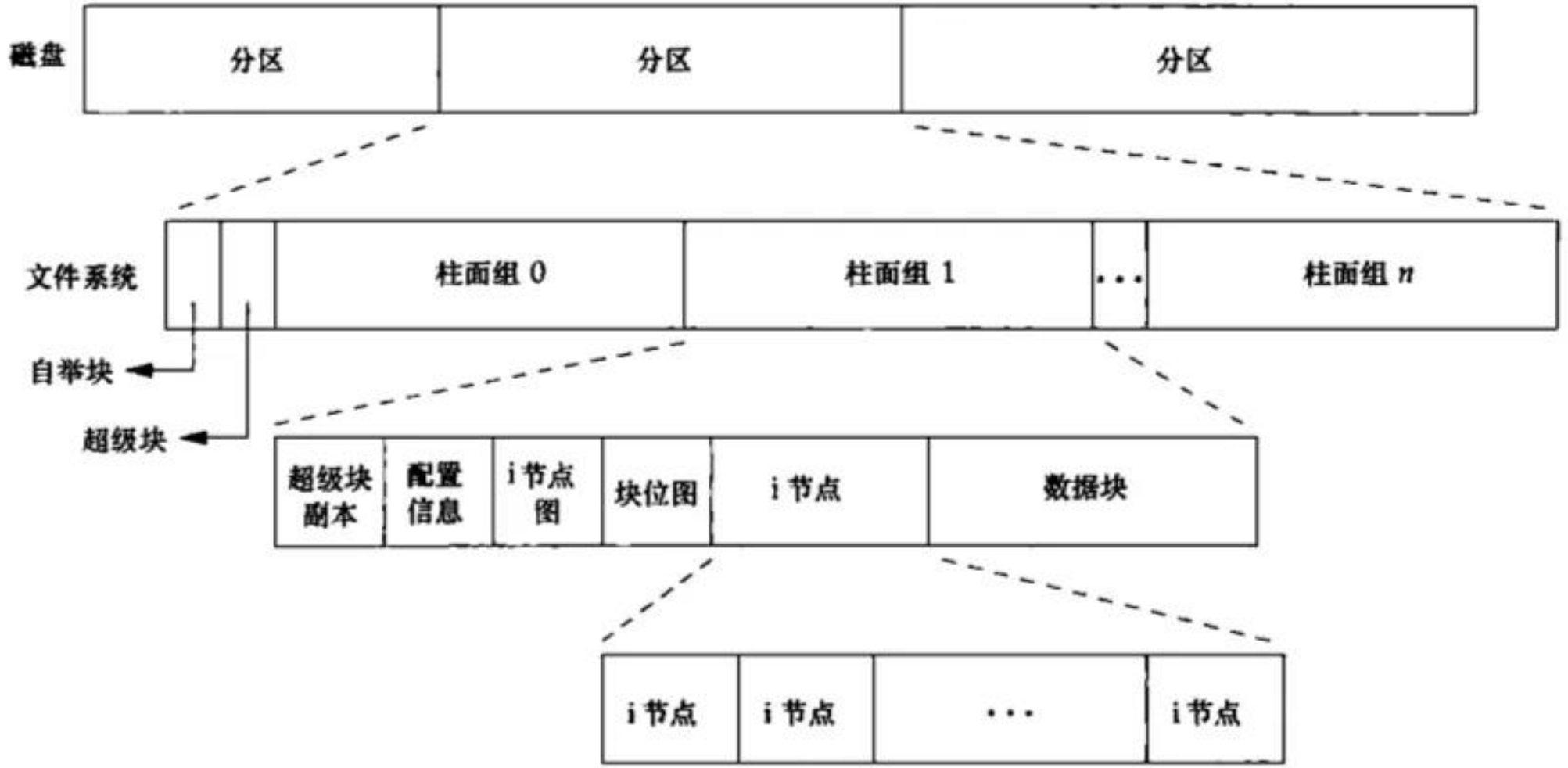


图 4-13　磁盘、分区和文件系统

文件

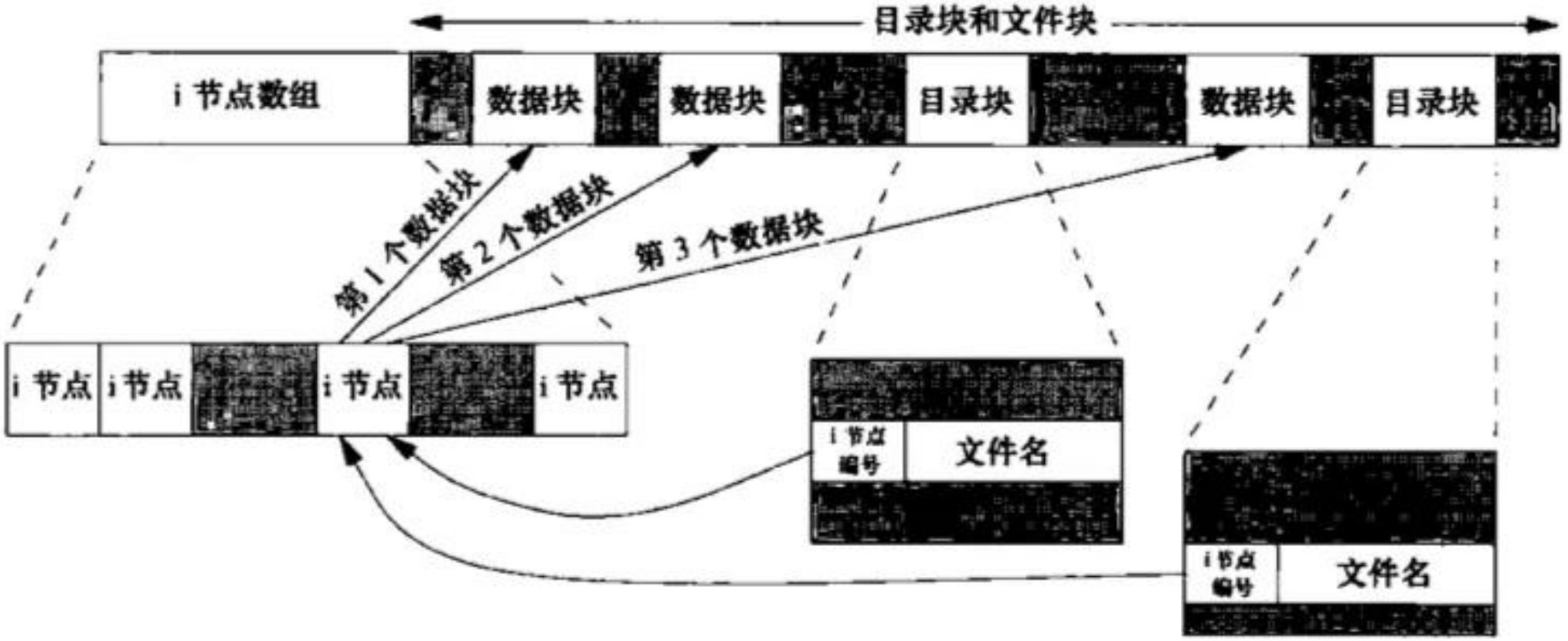


图 4-14　较详细的柱面组的 i 节点和数据块

目录

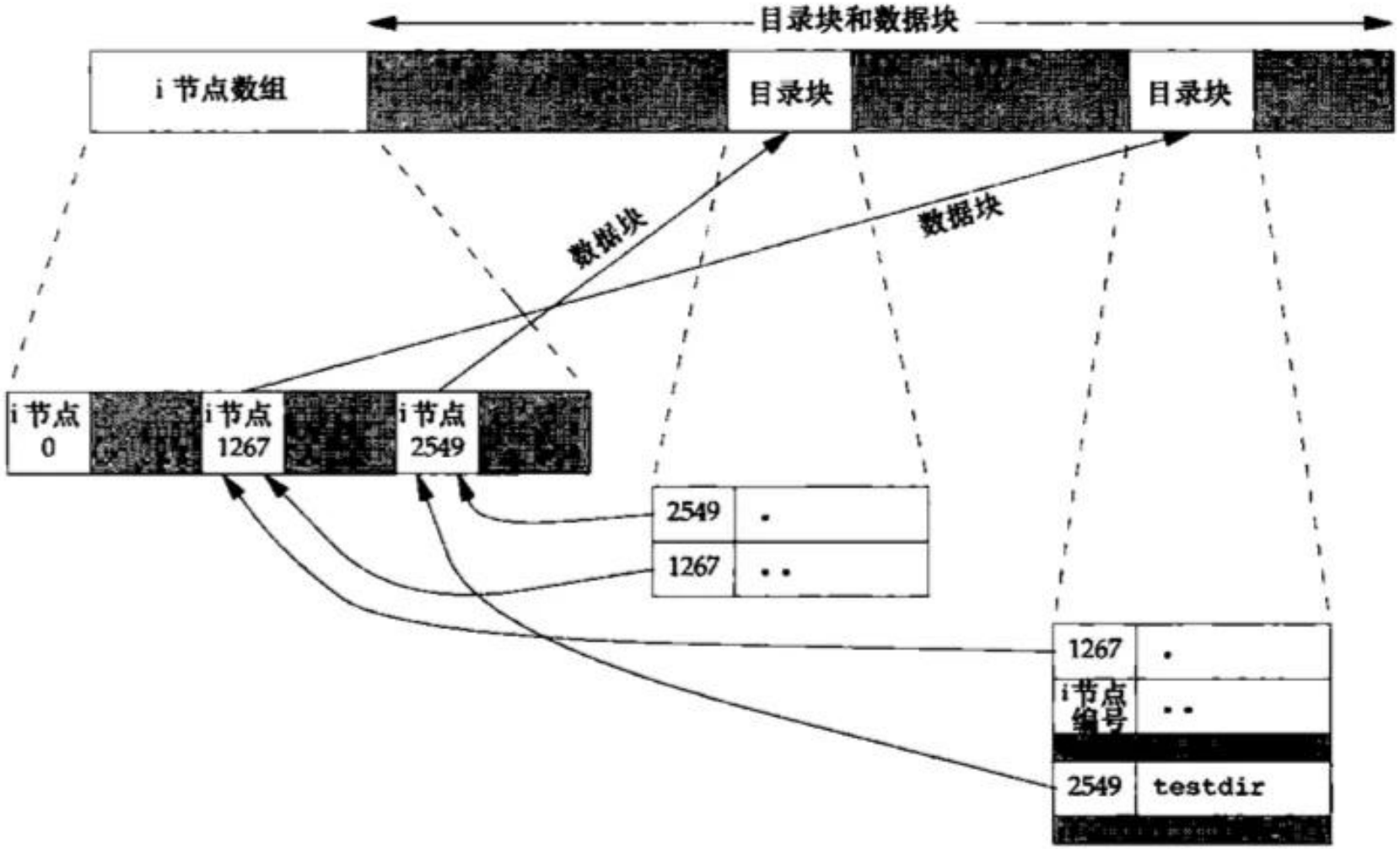


图 4-15　创建了目录 testdir 后的文件系统实例

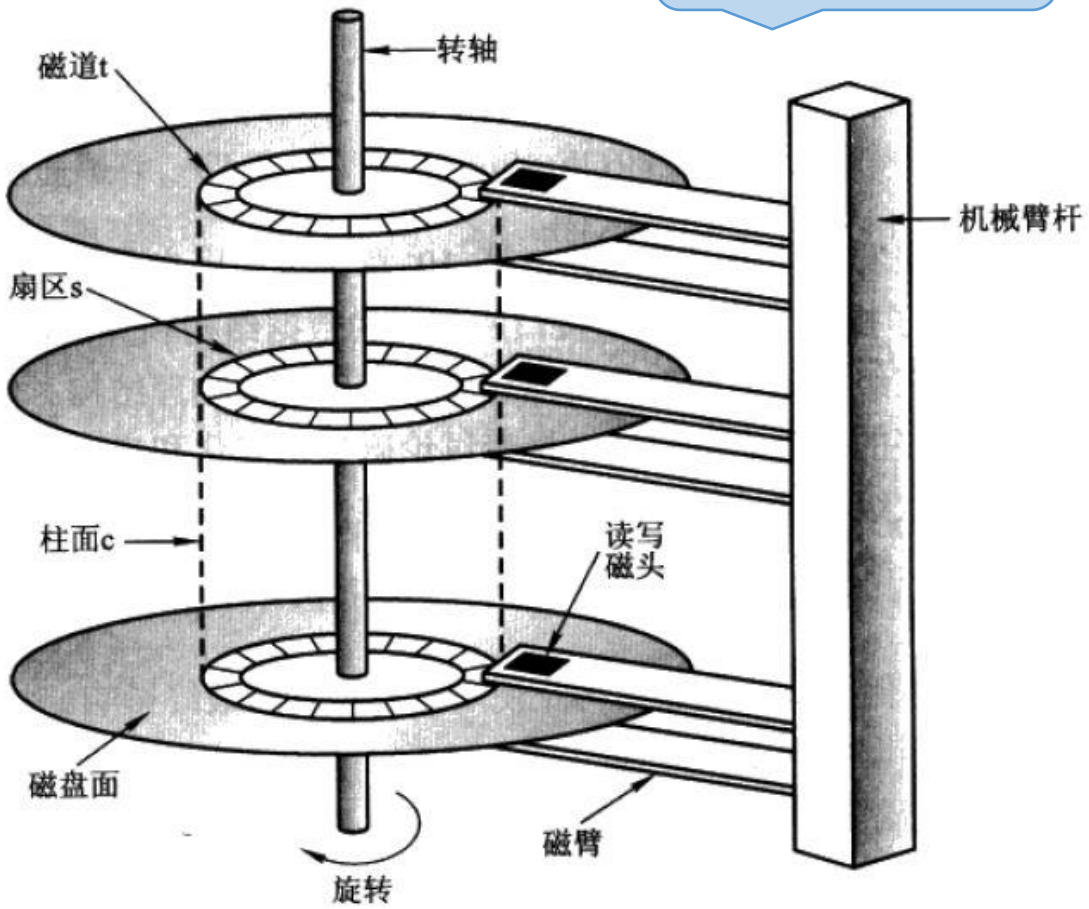
文件查找

例子： /home/jinxin/kernel.c
（１）找到 / 的inode，比如是 2 。
由于inode 2是目录类型，所以inode 2指向的**数据块**里保存的都是dentry(也就是目录项)
（２）找到home对应的inode,比如是12。
由于inode 12也是目录类型，所以inode 12指向的**数据块**里保存的也是dentry
（３）找到jinxin对应的inode,比如是112。
由于inode 112也是目录类型，所以inode 112指向的**数据块**里保存的也是dentry
（４）找到kernel.c对应的inode,比如是1112。
由于inode 1112是文件类型，所以到这里就寻找结束了。
下一步如果需要读取数据，可以从inode 1112指向的数据块里读取文件内容。

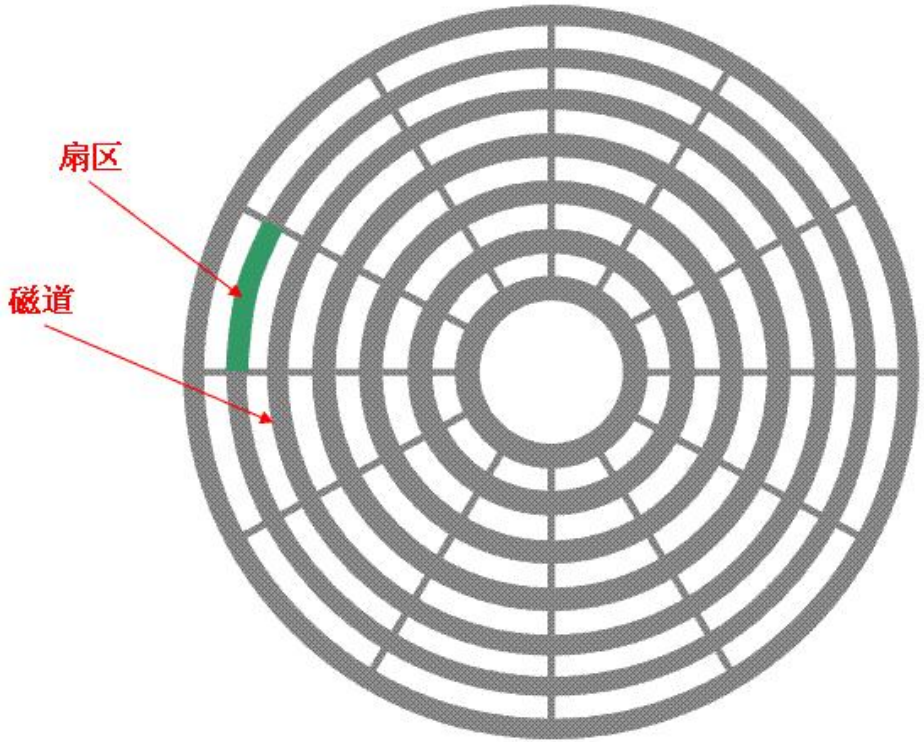
为加速文件和目录的查找 → Dentry Cache(Dcache)

HDD结构

磁头数=盘面数

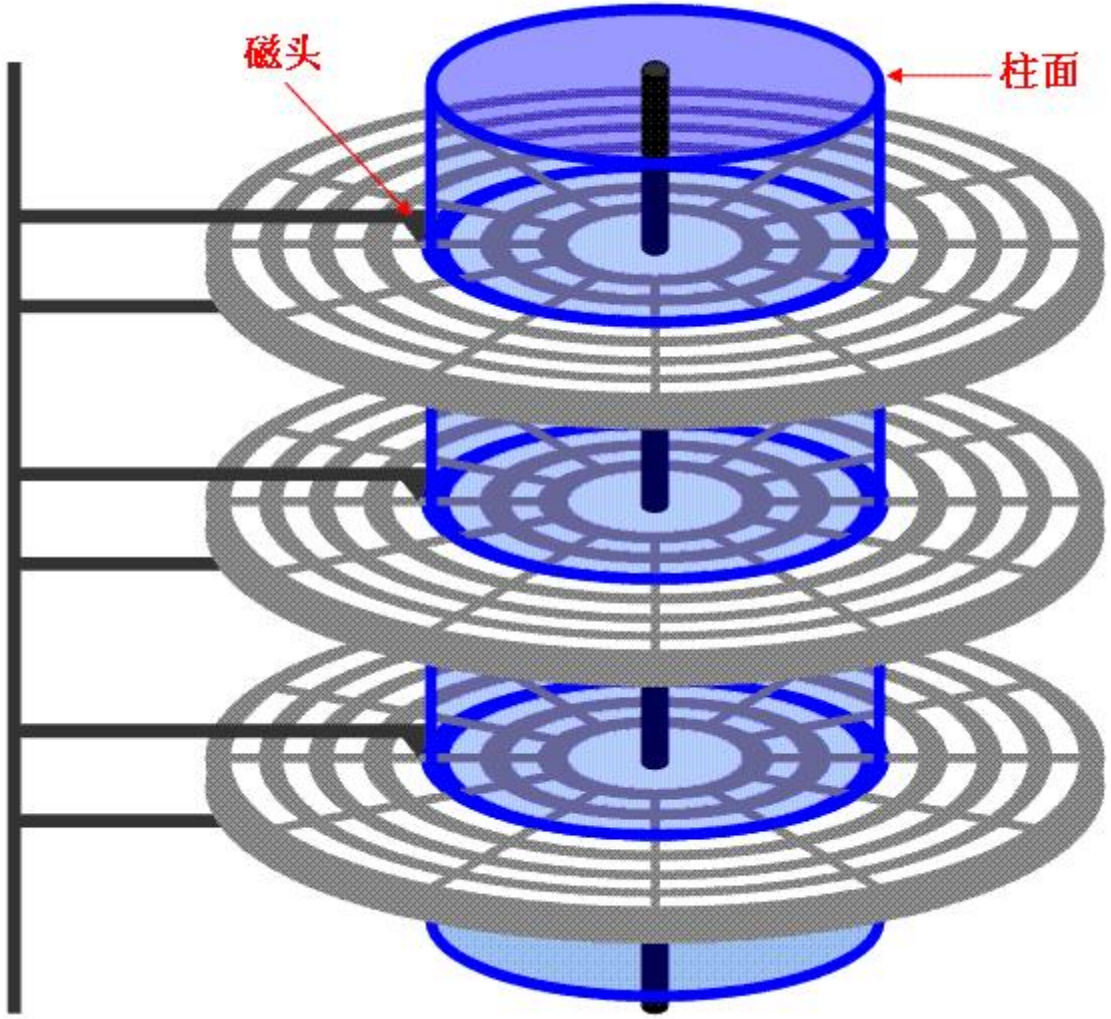


一个磁道包
含多个扇面



存储容量 = 磁头数 × 磁道数 × 每道扇区数 × 每扇区字节数

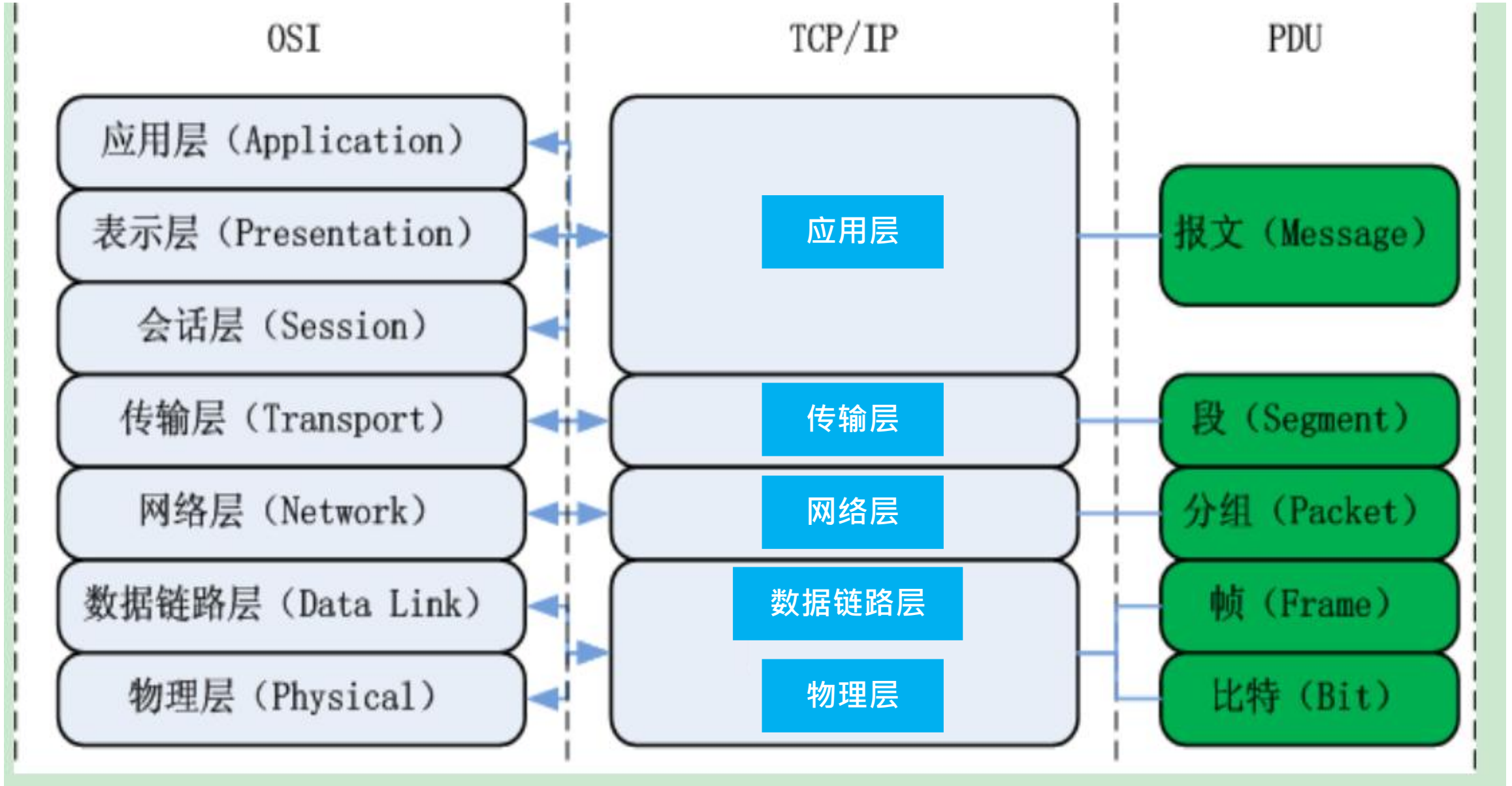
HDD读取时间



磁盘读取响应时间：

1. 寻道时间
2. 旋转延迟
3. 数据传输时间

TCP/IP网络模型

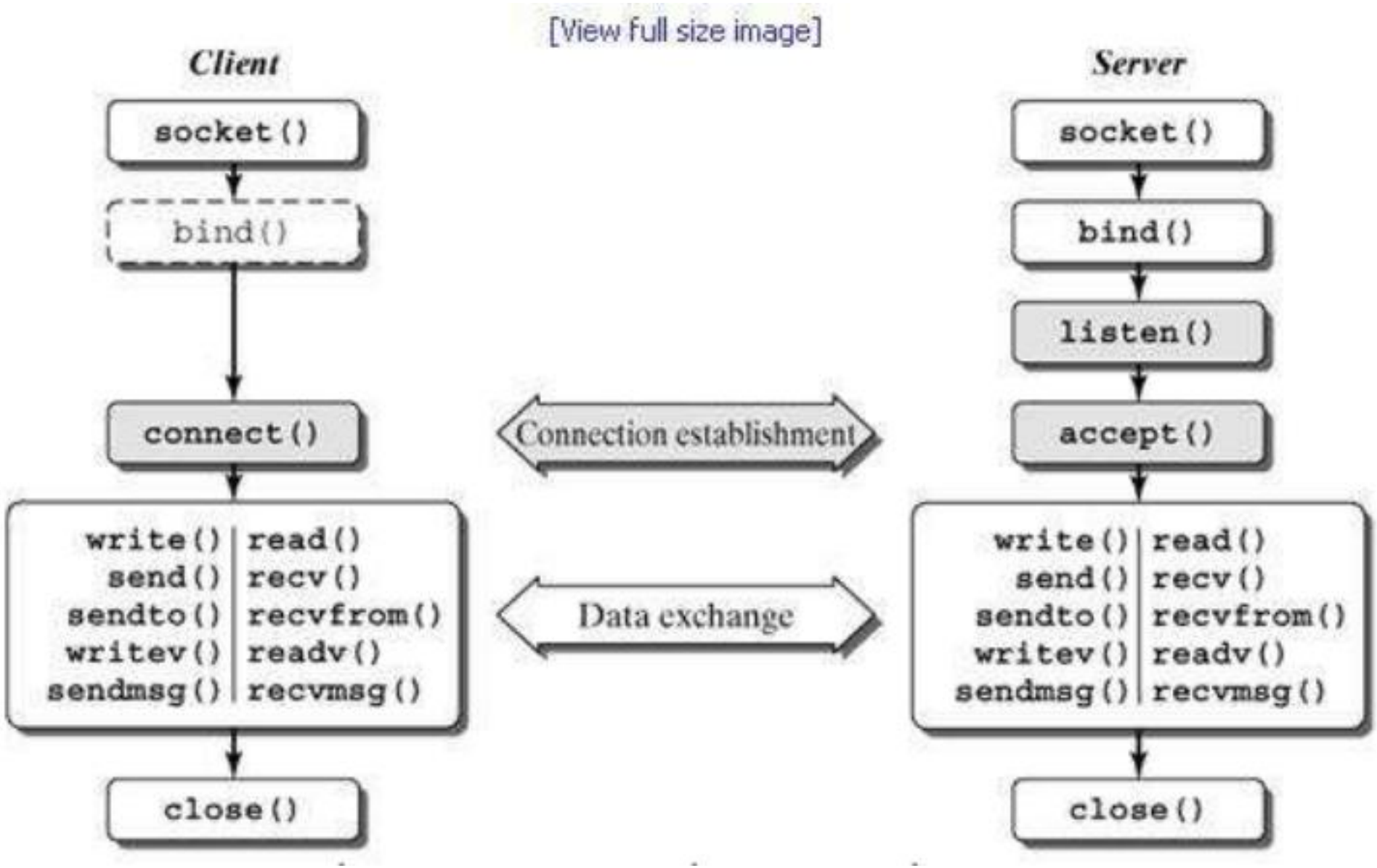

OSI
TCP/IP
PDU
应用层（Application）
表示层（Presentation）
会话层（Session）
传输层（Transport）
网络层（Network）
数据链路层（Data Link）
物理层（Physical）
应用层
传输层
网络层
数据链路层
物理层
报文（Message）
段（Segment）
分组（Packet）
帧（Frame）
比特（Bit）

网络模型和协议

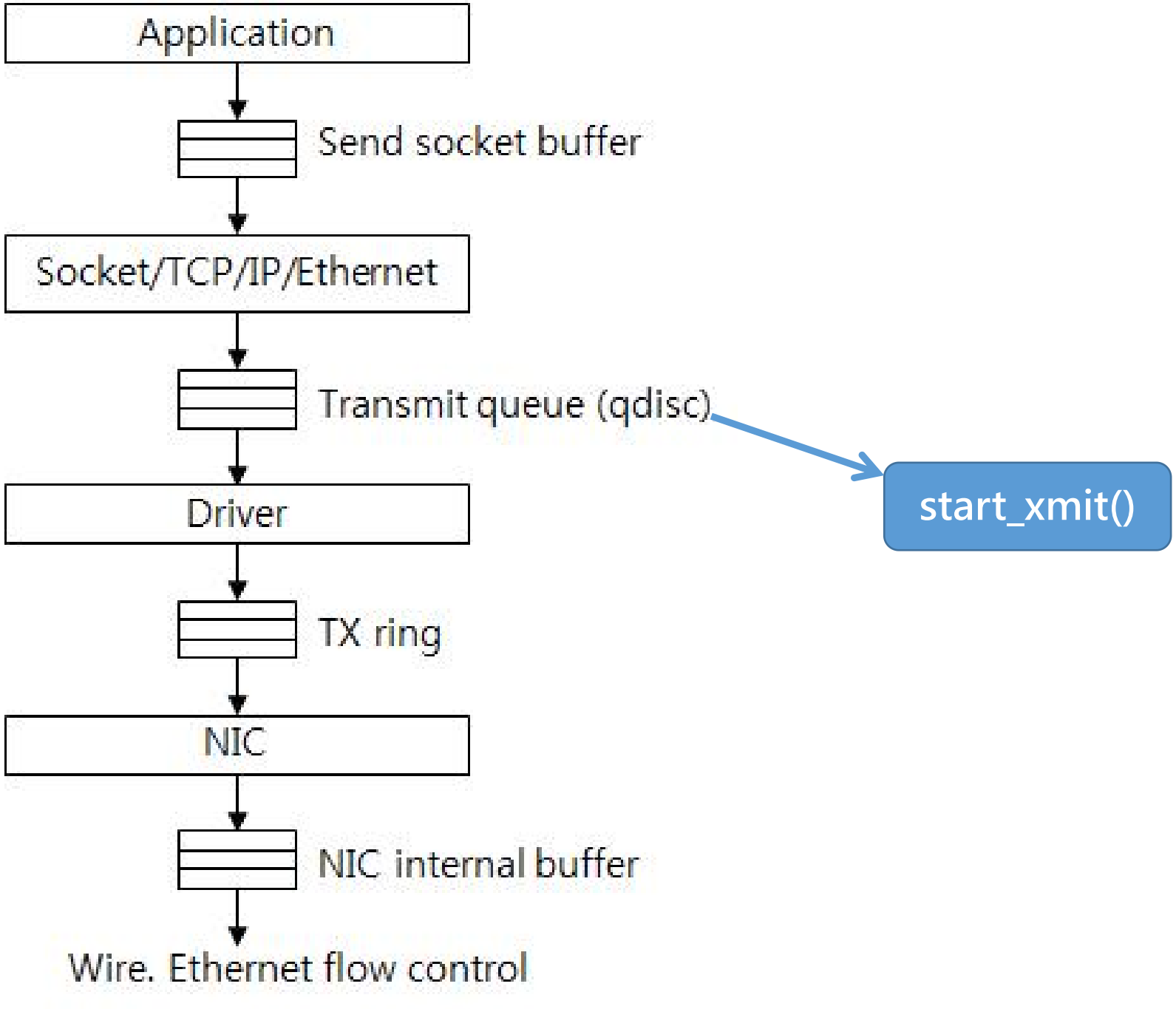
OSI七层网络模型	TCP/IP四层概念模型	对应网络协议
应用层（Application）	应用层	HTTP、TFTP, FTP, NFS, WAIS、SMTP
表示层（Presentation）		Telnet, Rlogin, SNMP, Gopher
会话层（Session）		SMTP, DNS
传输层（Transport）	传输层	TCP, UDP
网络层（Network）	网络层	IP, ICMP, ARP, RARP, AKP, UUCP
数据链路层（Data Link）	数据链路层	FDDI, Ethernet, Arpanet, PDN, SLIP, PPP
物理层（Physical）		IEEE 802.1A, IEEE 802.2到IEEE 802.11


应用层
数据
传输层
TCP 包首部
数据
TCP 中的数据
网络层
IP 包首部
TCP 包首部
数据
IP 中的数据
数据链路层
以太网包首部
IP 包首部
TCP 包首部
数据
以太网中的数据

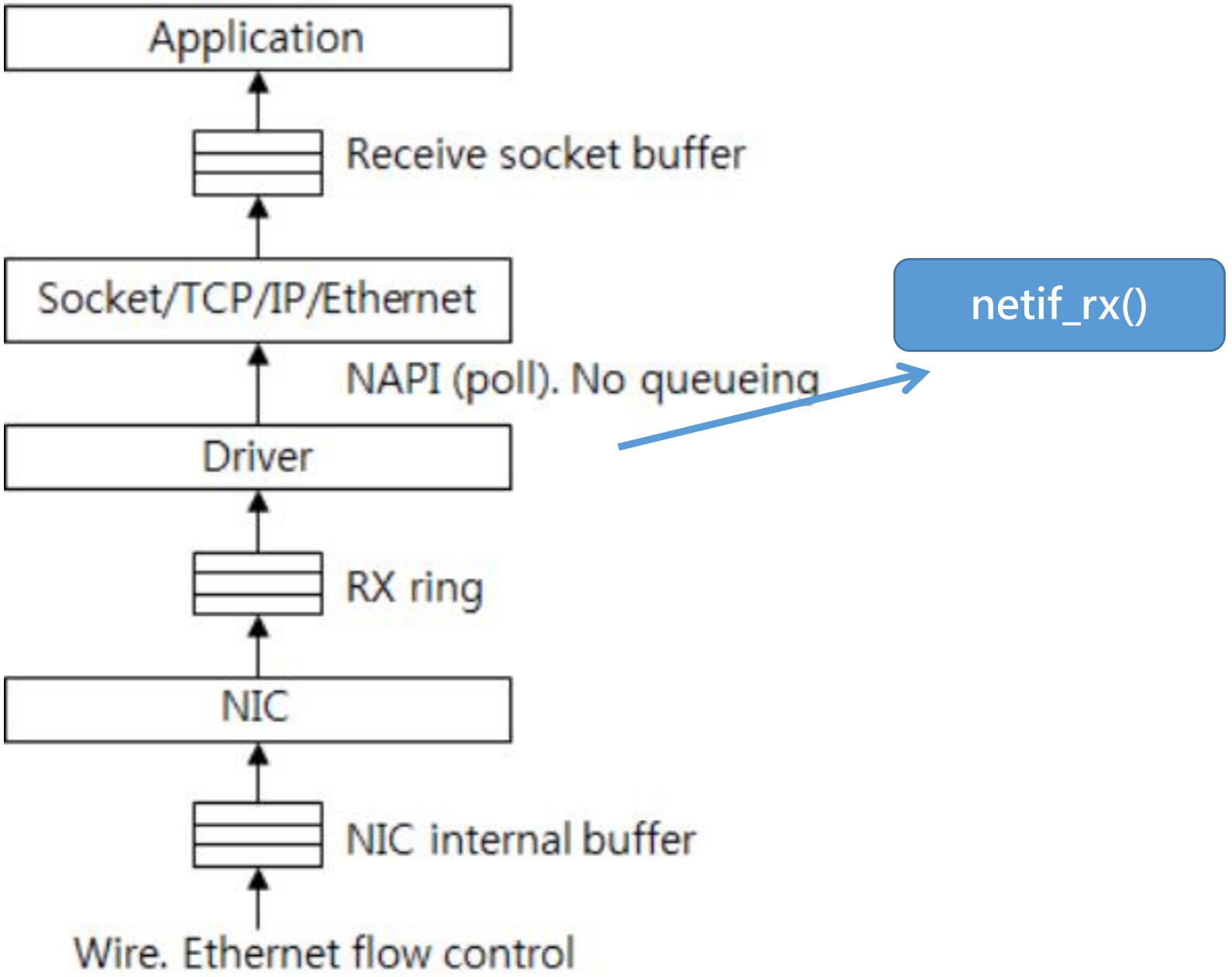
server-client



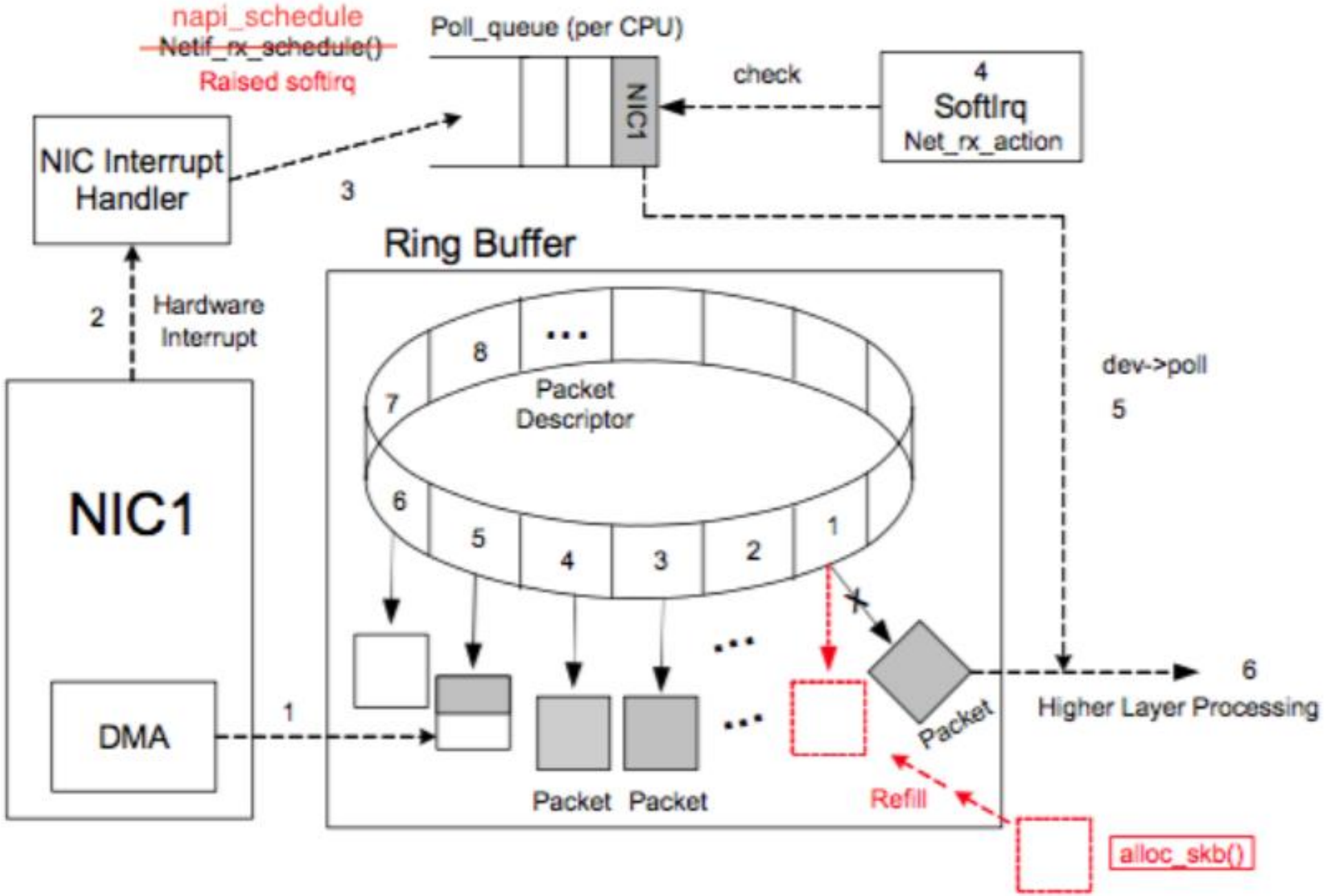
数据发送流程


Application
Send socket buffer
Socket/TCP/IP/Ethernet
Transmit queue (qdisc)
start_xmit()
Driver
TX ring
NIC
NIC internal buffer
Wire. Ethernet flow control

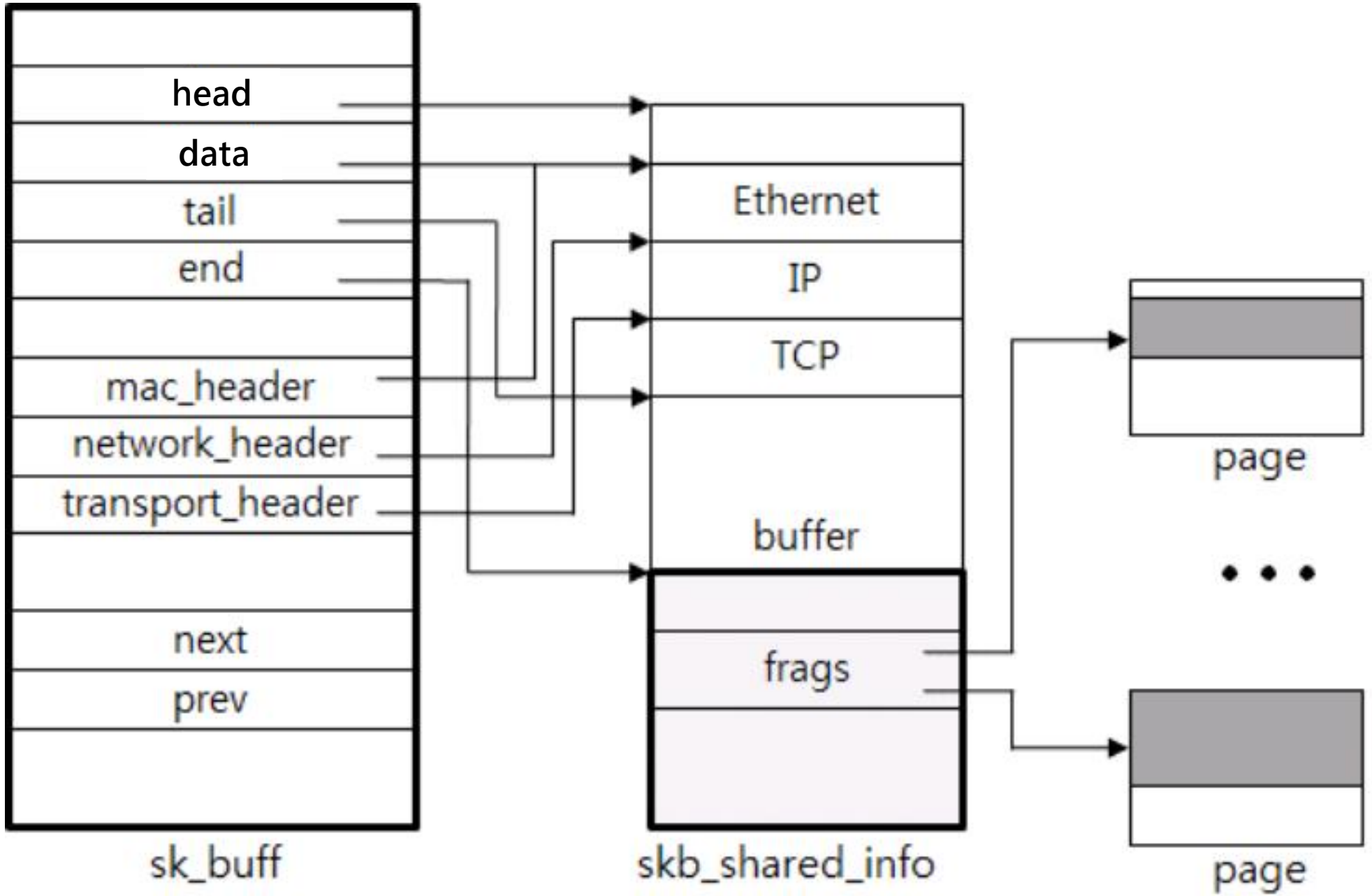
数据接收流程

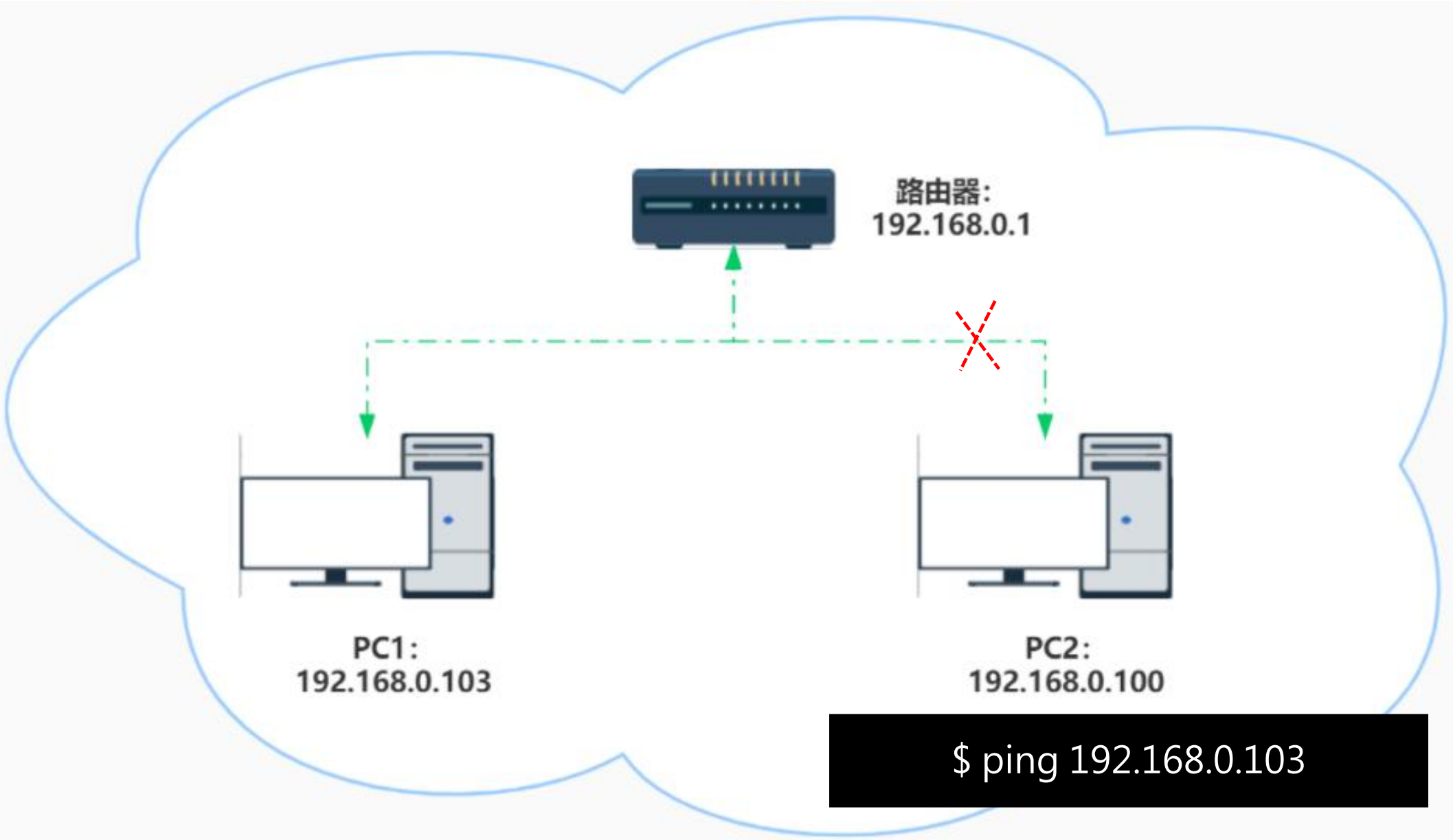

Application
Receive socket buffer
Socket/TCP/IP/Ethernet
NAPI (poll). No queueing
netif_rx()
Driver
RX ring
NIC
NIC internal buffer
Wire. Ethernet flow control

网卡驱动接收数据


napi_schedule
Netif_rx_schedule()
Raised softirq
Poll_queue (per CPU)
NIC1
check
4
SoftIrq
Net_rx_action
NIC Interrupt Handler
3
Ring Buffer
2
Hardware Interrupt
dev->poll
5
NIC1
8
7
6
5
4
3
2
1
Packet Descriptor
DMA
1
Packet
Packet
Packet
6
Higher Layer Processing
Refill
alloc_skb()

sk_buff


head
data
tail
end
mac_header
network_header
transport_header
next
prev
sk_buff
Ethernet
IP
TCP
buffer
frags
skb_shared_info
page
page


路由器:
192.168.0.1
PC1:
192.168.0.103
PC2:
192.168.0.100
$ ping 192.168.0.103

Host A
192.168.0.100

NGINX

Host B
192.168.119.141

$ ab -c 10 -n 1000 -k
http://192.168.0.100:8081/

改一行配置就能
解决问题？？？