

Linux 铁三角之I/O(四)

讲解时间：4月15日、4月16日、4月17日、4月18日晚9点

宋宝华 <21cnbao@gmail.com>

微信群直播：

https://mp.weixin.qq.com/s/hi5Xt_dZmtDZZHoqZeyjdA

扫描二维码报名



Linux任督二脉

(学习形式：微信群)



麦当劳喜欢您来，喜欢您再来



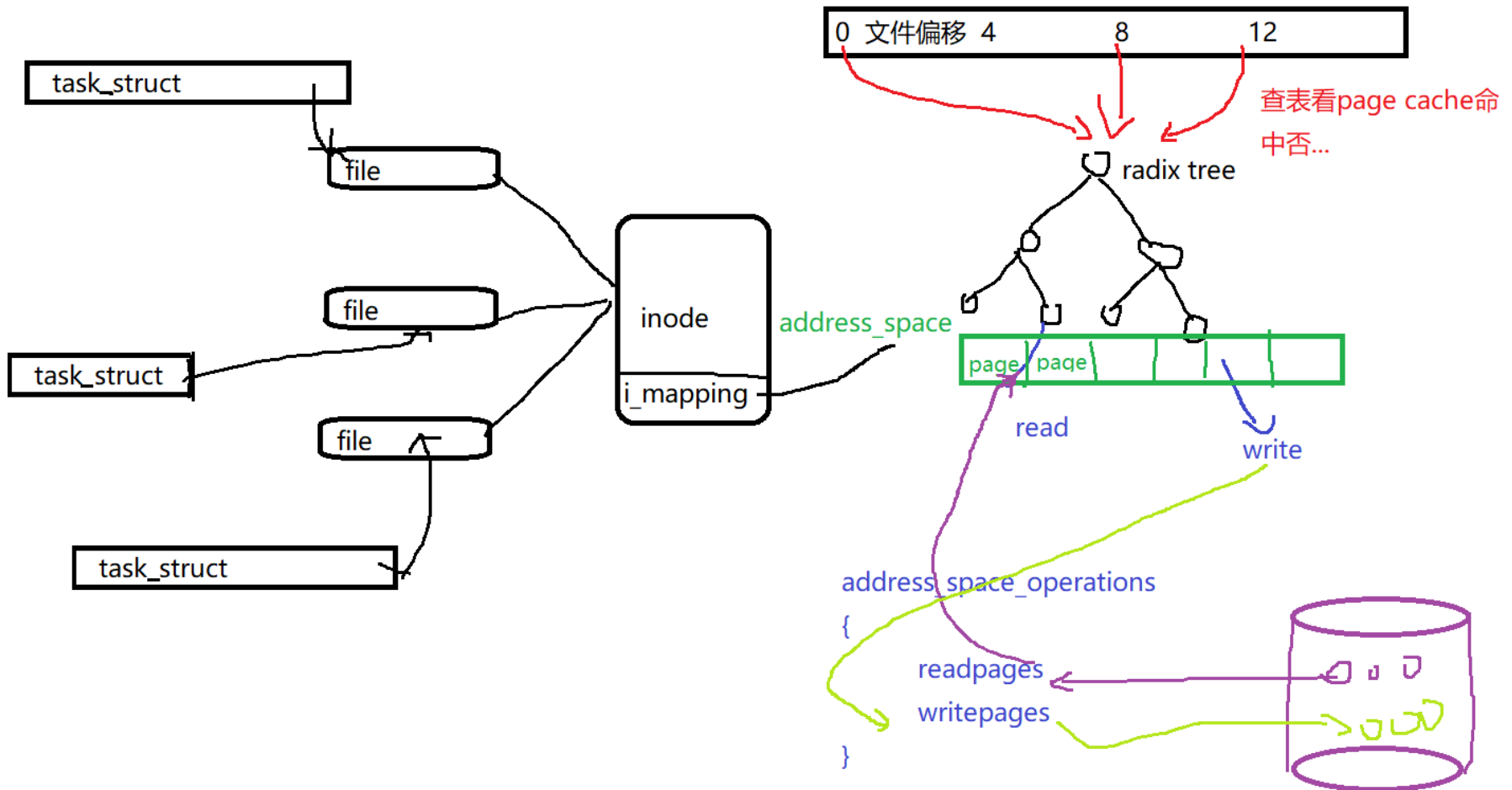
扫描关注
Linuxer



块I/O流程与I/O调度器

- *一个块IO的一生：从page cache到bio到request
- *O_DIRECT和O_SYNC
- *blktrace
- *IO调度和CFQ调度算法
- *CFQ和ionice
- *cgroup与IO
- *io性能调试：iotop, iostat

file 到 address_space_operations



i_mapping 决定是buffers还是cached

```
void si_meminfo(struct sysinfo *val)
```

```
{
```

```
    val->totalram = totalram_pages;
```

```
    val->sharedram = global_page_state(NR_SHMEM);
```

```
    val->freeram = global_page_state(NR_FREE_PAGES);
```

```
    val->bufferram = nr_blockdev_pages();
```

```
    val->totalhigh = totalhigh_pages;
```

```
    val->freehigh = nr_free_highpages();
```

```
    val->mem_unit = PAGE_SIZE;
```

```
}
```

```
long nr_blockdev_pages(void)
```

```
{
```

```
    struct block_device *bdev;
```

```
    long ret = 0;
```

```
    spin_lock(&bdev_lock);
```

```
    list_for_each_entry(bdev, &all_bdevs, bd_list) {
```

```
        ret += bdev->bd_inode->i_mapping->npages;
```

```
    }
```

```
    spin_unlock(&bdev_lock);
```

```
    return ret;
```

```
}
```

/mnt/a/b.main

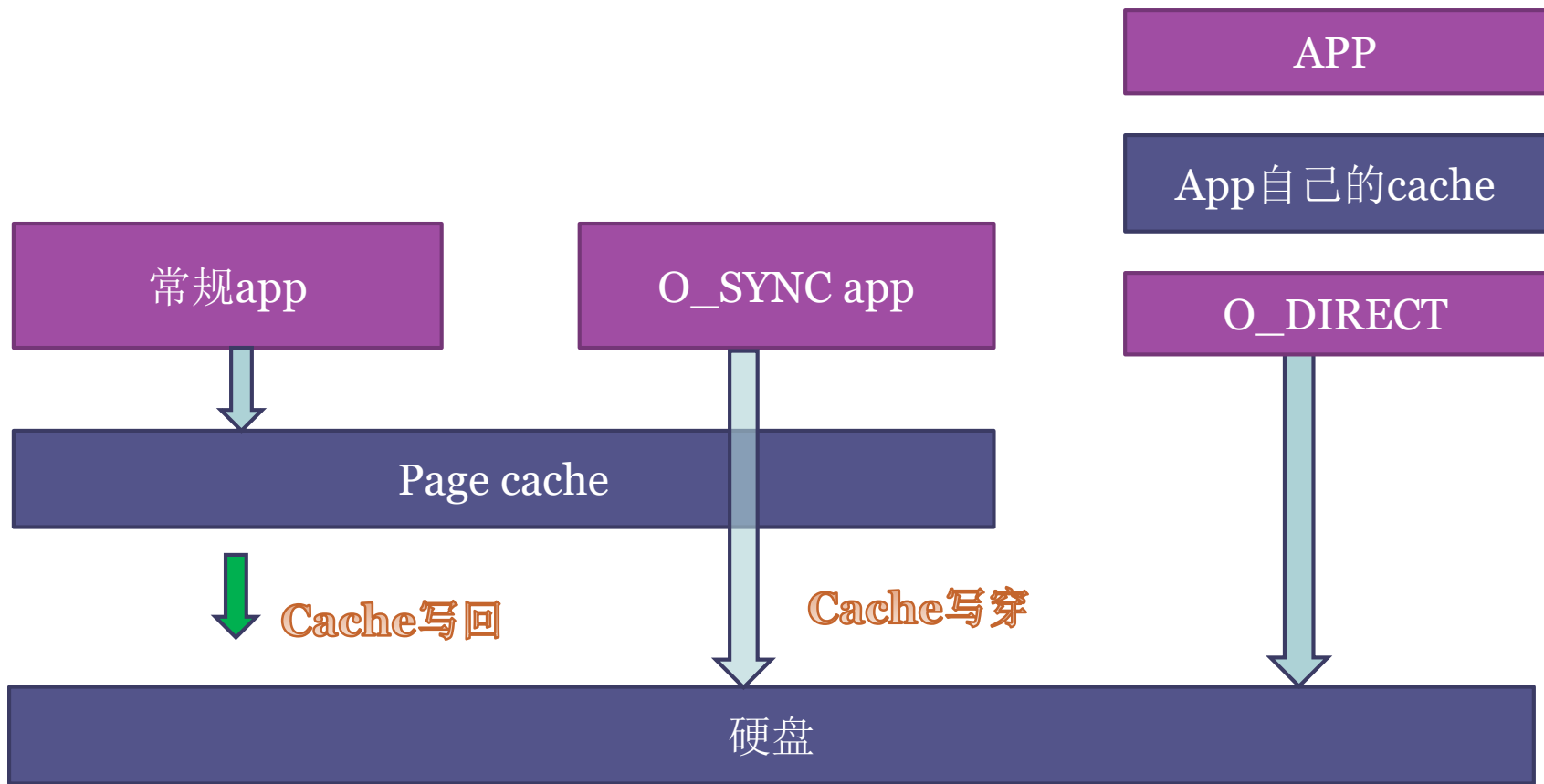
/dev/sda1

```
baohua@baohua-VirtualBox:~/develop/linux$ free
```

	total	used	free	shared
Mem:	1024844	815348	209496	2060
-/+ buffers/cache:		277704	747140	
Swap:	522236	235132	287104	

buffers	cached
30440	507204

O_DIRECT 和 O_SYNC



"The thing that has always disturbed me about O_DIRECT is that the whole interface is just stupid, and was probably designed by a deranged monkey on some serious mind-controlling substances."—Linus

Block IO 流程

app



文件系统

submit_bio()



bio

io调度
make_request_fn()



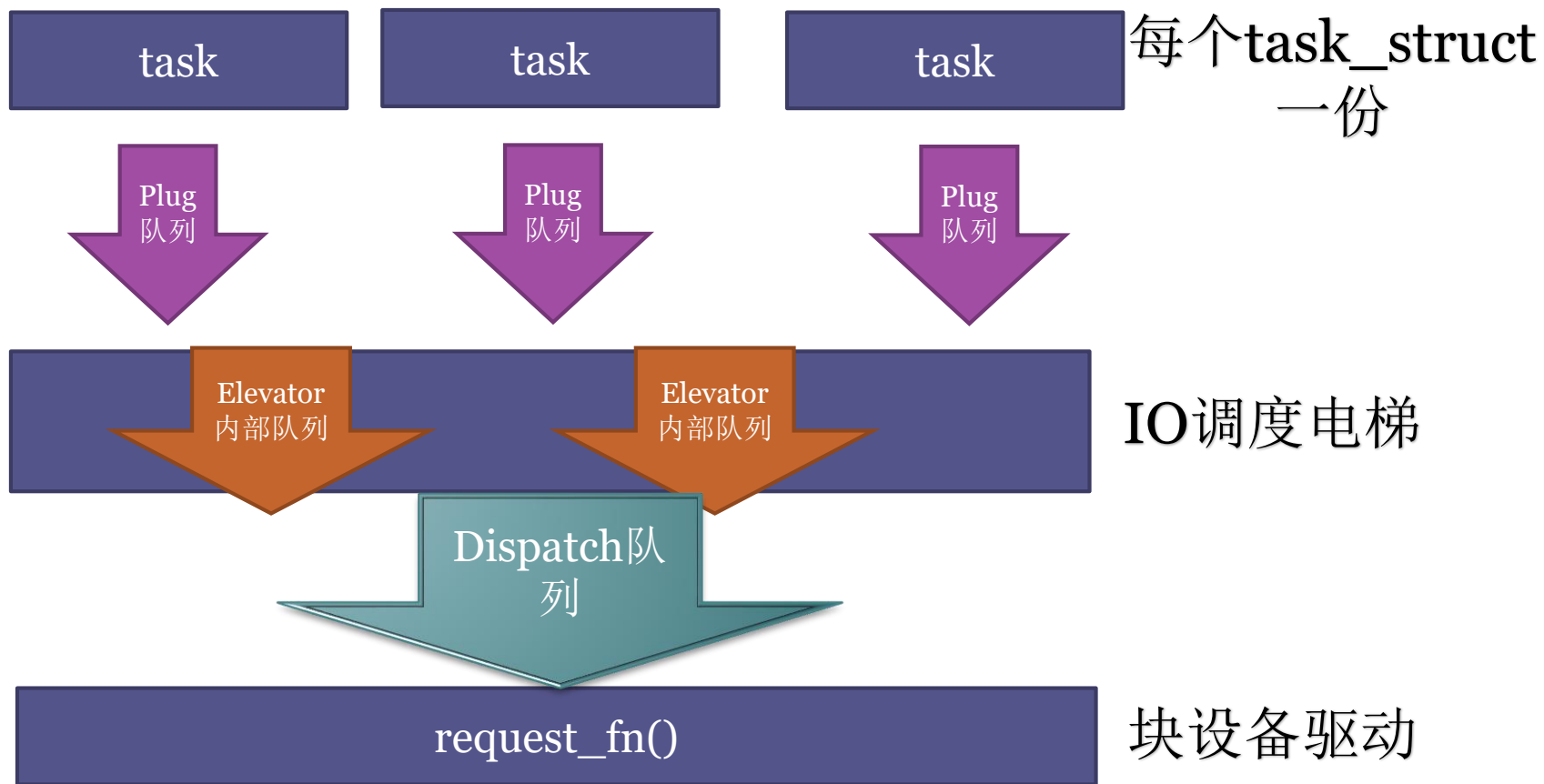
request

块设备驱动

合并的: elv_merge

新的get_request

Request的三个队列



读写流程的ftrace

```
#include <unistd.h>
#include <fcntl.h>

main()
{
    int fd;
    char buf[4096];

    sleep(30); //run ./funtion.sh to trace this process
    fd = open("file", O_RDONLY);
    read(fd, buf, 4096);
    read(fd, buf, 4096);
}
```

```
#!/bin/bash
```

```
debugfs=/sys/kernel/debug
echo nop > $debugfs/tracing/current_tracer
echo o > $debugfs/tracing/tracing_on
echo `pidof read` > $debugfs/tracing/set_ftrace_pid
echo function_graph > $debugfs/tracing/current_tracer
echo vfs_read > $debugfs/tracing/set_graph_function
echo 1 > $debugfs/tracing/tracing_on
```

查看 ftrace 结果

- # cat /sys/kernel/debug/tracing/trace

0)		read_pages() {
0)	0.068 us	blk_start_plug();
0)		ext4_readpages() {
0)		mpage_readpages() {
0)		add_to_page_cache_lru() {
0)		__add_to_page_cache_locked() {
0)	0.055 us	PageHuge();
0)		mem_cgroup_try_charge() {
0)	0.384 us	get_mem_cgroup_from_mm();
0)	0.169 us	try_charge();
0)	1.391 us	}
0)	0.060 us	_raw_spin_lock_irq();
0)	0.181 us	page_cache_tree_insert();
0)		__inc_zone_page_state() {
0)	0.053 us	__inc_zone_state();
0)	0.536 us	}
0)		mem_cgroup_commit_charge() {
0)	0.126 us	mem_cgroup_charge_statistics.isra.32();
0)	0.052 us	memcg_check_events();
0)	1.007 us	}
0)	5.690 us	}
0)		lru_cache_add() {
0)	0.123 us	__lru_cache_add();
0)	0.962 us	}
0)	7.399 us	}
0)		do_mpage_readpage() {

blktrace

```
baohua@baohua-VirtualBox:~/develop/linux$ sudo blktrace -d /dev/sda -o - |blkparse -i -  
[sudo] password for baohua:
```

设备号 Major:minor	CPU	序号	时间戳	PID	事件	开始块+块数	进程名
8,0	0	1	0.000000000	167	A WS	12957520 + 8	<- (8,1) 12955472
8,0	0	2	0.000002319	167	Q WS	12957520 + 8	[jbd2/sda1-8]
8,0	0	3	0.000007055	167	G WS	12957520 + 8	[jbd2/sda1-8]
8,0	0	4	0.000008384	167	P N	[jbd2/sda1-8]	
8,0	0	5	0.000010005	167	A WS	12957528 + 8	<- (8,1) 12955480
8,0	0	6	0.000010513	167	Q WS	12957528 + 8	[jbd2/sda1-8]
8,0	0	7	0.000012040	167	M WS	12957528 + 8	[jbd2/sda1-8]
8,0	0	8	0.000013173	167	A WS	12957536 + 8	<- (8,1) 12955488
8,0	0	9	0.000013638	167	Q WS	12957536 + 8	[jbd2/sda1-8]
8,0	0	10	0.000014240	167	M WS	12957536 + 8	[jbd2/sda1-8]
8,0	0	11	0.000015133	167	A WS	12957544 + 8	<- (8,1) 12955496
8,0	0	12	0.000015596	167	Q WS	12957544 + 8	[jbd2/sda1-8]
8,0	0	13	0.000016182	167	M WS	12957544 + 8	[jbd2/sda1-8]
8,0	0	14	0.000016963	167	A WS	12957552 + 8	<- (8,1) 12955504
8,0	0	15	0.000017427	167	Q WS	12957552 + 8	[jbd2/sda1-8]
8,0	0	16	0.000018013	167	M WS	12957552 + 8	[jbd2/sda1-8]
8,0	0	17	0.000019305	167	A WS	12957560 + 8	<- (8,1) 12955512
8,0	0	18	0.000019772	167	Q WS	12957560 + 8	[jbd2/sda1-8]
8,0	0	19	0.000020361	167	M WS	12957560 + 8	[jbd2/sda1-8]
8,0	0	20	0.000021561	167	A WS	12957568 + 8	<- (8,1) 12955520
8,0	0	21	0.000022025	167	Q WS	12957568 + 8	[jbd2/sda1-8]
8,0	0	22	0.000022608	167	M WS	12957568 + 8	[jbd2/sda1-8]
8,0	0	23	0.000023381	167	A WS	12957576 + 8	<- (8,1) 12955528

读写, sync, barrier

IO 调度算法

```
baohua@baohua-VirtualBox:/sys/block/sda/queue$ cat scheduler  
noop [deadline] cfq
```

- ✓ Noop: 最简单的调度器，把邻近bio进行了合并处理。
- ✓ Deadline: 保证读优先的前提下，写不会饿死。
- ✓ CFQ: 考虑进程。

CFQ和ionice

```
root@baohua-VirtualBox:/sys/block/sda/queue# echo cfq > scheduler
```

```
# ionice -c 2 -n 0 cat /dev/sda > /dev/null&
```

```
[1] 7392
```

```
# ionice -c 2 -n 7 cat /dev/sda > /dev/null&
```

```
[2] 7393
```

iotop的结果

Total DISK READ :	444.44 M/s	Total DISK WRITE :	0.00 B/s				
Actual DISK READ:	444.68 M/s	Actual DISK WRITE:	0.00 B/s				
TID	PRIO	USER	DISK READ	DISK WRITE	SWAPIN	IO>	COMMAND
7393	be/7	root	72.91 M/s	0.00 B/s	0.00 %	99.99 %	cat /dev/sda
7392	be/0	root	371.53 M/s	0.00 B/s	0.00 %	81.45 %	cat /dev/sda
1	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	init
2	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kthreadd]
3	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[ksoftirqd/0]
5	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kworker/0:0H]
7	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[rcu_sched]
8	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[rcu_bh]
9	rt/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[migration/0]
10	rt/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[watchdog/0]
11	rt/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[watchdog/1]

至于 cgroup 的 weight 和 throttle

```
mkdir -p /sys/fs/cgroup/blkio/A/ /sys/fs/cgroup/blkio/B
```

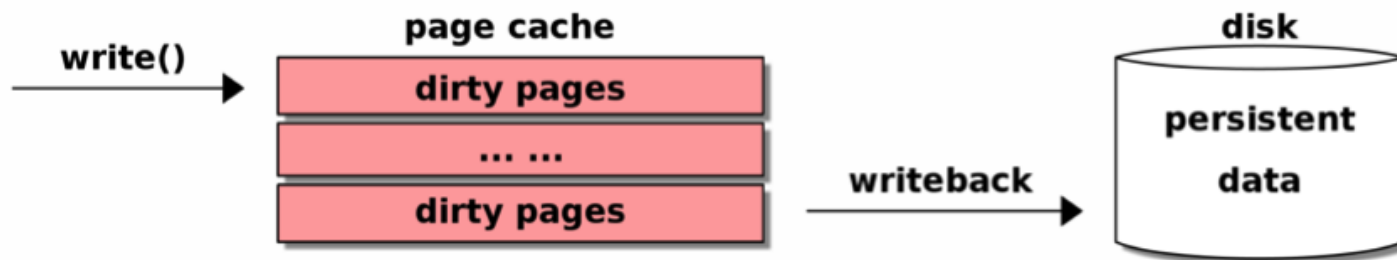
```
cgexec -g blkio:A dd if=/dev/sda of=/dev/null &  
cgexec -g blkio:B dd if=/dev/sda of=/dev/null &
```

```
cgexec -g blkio:A dd if=/dev/zero of=/mnt/a oflag=direct bs=1M count=300  
&
```

```
echo "8:o 1048576" >  
/sys/fs/cgroup/blkio/A/blkio.throttle.read_bps_device  
echo "8:o 1048576" >  
/sys/fs/cgroup/blkio/A/blkio.throttle.write_bps_device
```

Cgroup v2的writeback throttle

在Cgroup v1里面，blkio cgroup的写只能用于DIRECT_IO的场景(writeback的线程和写的不是一个线程)。
这使得write变成了“system wide”而不是group wide



在Cgroup v2里面，打通了memory group和blkio group，能知晓每个group的dirty情况。

iostat

```
# ionice -c 2 -n 0 cat /dev/sda > /dev/null&
# ionice -c 2 -n 7 cat /dev/sda > /dev/null&
```

```
root@baohua-VirtualBox:/sys/block/sda/queue# iostat -txz 1
```

```
Linux 4.0.0-040000-generic (baohua-VirtualBox) 2018年03月10日 _i686_ (4 CPU)
```

2018年03月10日 19时00分18秒

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.62    0.01    0.08    0.07    0.00   99.22
```

Device:	rrqm/s	wrqm/s	r/s	w/s	rkB/s	wkB/s	avgrq-sz	avgqu-sz	await	r_await	w_await	svctm	%util
loop0	0.00	0.00	0.00	0.00	0.01	0.00	8.00	0.00	0.40	0.40	0.00	0.40	0.00
loop1	0.00	0.00	0.00	0.00	0.01	0.00	7.83	0.00	0.54	0.43	4.00	0.54	0.00
sda	0.04	0.62	5.02	0.16	947.90	5.66	368.15	0.01	1.43	1.37	3.37	0.49	0.25

2018年03月10日 19时00分19秒

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           9.19    0.00    4.46   33.07    0.00   53.28
```

Device:	rrqm/s	wrqm/s	r/s	w/s	rkB/s	wkB/s	avgrq-sz	avgqu-sz	await	r_await	w_await	svctm	%util
sda	27.00	6832.00	898.00	50.00	235608.00	27528.00	555.14	2.89	3.08	1.72	27.44	1.03	98.00

2018年03月10日 19时00分20秒

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          10.67    0.00    2.67   25.33    0.00   61.33
```

Device:	rrqm/s	wrqm/s	r/s	w/s	rkB/s	wkB/s	avgrq-sz	avgqu-sz	await	r_await	w_await	svctm	%util
sda	29.00	6090.00	1145.00	47.00	292744.00	22064.00	528.20	3.87	3.18	1.84	35.74	0.82	97.60

谢谢！