OpenEuler

IO性能分析与问题定位

目录

CONTENT

OpenEuler

iostat

原理介绍

- 内核结构体，在每个设备上记录io的相关信息

```
enum stat_group {
        STAT_READ,
        STAT_WRITE,
        STAT_DISCARD,
        STAT_FLUSH,

        NR_STAT_GROUPS
};
struct disk_stats {
        u64 nsecs[NR_STAT_GROUPS];
        unsigned long sectors[NR_STAT_GROUPS];
        unsigned long ios[NR_STAT_GROUPS];
        unsigned long merges[NR_STAT_GROUPS];
        unsigned long io_ticks;
        local_t in_flight[2];
};
```

- 内核提供sysfs接口打印存储的信息，一共17个字段
  - 文件名：　　stat
  - 内核接口：　　part_stat_show()

iostat

原理介绍

- 通过sysfs接口读取信息，计算并打印

字段	含义	内核统计接口	iostat	含义
1	读io数量	part_stat_inc(part, ios[STAT_READ])	r/s	每秒完成读请求数量
2	读io合并数量	part_stat_inc(part, merges[STAT_READ])	rrqm/s	读请求每秒合并次数
3	读io扇区数量	part_stat_add(part, sectors[STAT_READ], sectors)	rkB/s	每秒读的带宽
4	读io花费时间总和	part_stat_add(part, nsecs[STAT_READ], duration)	r_await	读请求平均等待时间
5	写io数量	part_stat_inc(part, ios[STAT_WRITE])	w/s	每秒完成写请求数量
6	写io合并数量	part_stat_inc(part, merges[STAT_WRITE])	wrqm/s	写请求每秒合并次数
7	写io扇区数量	part_stat_add(part, sectors[STAT_WRITE], sectors)	wkB/s	每秒写的带宽
8	写io花费时间总和	part_stat_add(part, nsecs[STAT_WRITE], duration)	w_await	写请求平均等待时间

iostat

原理介绍

- 通过sysfs接口读取信息

字段	含义	内核统计接口	iostat	含义
9	当前的inflight io数量	blk_mq_in_flight		
10	设备在处理io的时间	update_io_ticks	%util	磁盘利用率
11	所有io花费的时间总和	4, 8, 15, 17总和	aqu-sz	平均队列长度
12	discard io数量	part_stat_inc(part, ios[STAT_DISCARD])	d/s	每秒完成discard请求数量
13	discard io合并数量	part_stat_inc(part, merges[STAT_DISCARD])	drqm/s	discard请求每秒合并次数
14	discard io扇区数量	part_stat_add(part, sectors[STAT_DISCARD], sectors)	dkB/s	每秒discard的带宽
15	discard io花费时间总和	part_stat_add(part, nsecs[STAT_DISCARD], duration)	d_await	discard请求平均等待时间
16	flush io数量	part_stat_inc(part, ios[STAT_FLUSH])	f/s	每秒完成flush请求数量
17	flush io花费时间总和	part_stat_add(part, nsecs[STAT_FLUSH], duration)	f_await	flush请求平均等待时间

blktrace

流程

1. fd = open(“/dev/xxx”,O_RDONLY|O_NONBLOCK)
   - 打开块设备
2. ioctl(fd, BLKTRACESETUP, ...)
   - 在内核调用blk_trace_setup() 去注册tracepoint
3. ioctl(fd, BLKTRACESTART)
   - 标记开始
4. ioctl(fd, BLKTRACESTOP)
   - 标记结束，同一个设备无法并发
5. 利用debugfs接口获取信息
   - /sys/kernel/debug/block/xxx/tracexx (percpu)
6. loctl(fd, BLKTRACETEARDOWN)
   - 在内核调用blk_trace_remove()去删除注册的tracepoint

blktrace

相关tracepoint

action	含义	tracepoint
Q	开始新的io	trace_block_bio_queue
G	io生成request	trace_block_getrq
I	io到达调度器	trace_block_rq_insert
D	io下发到驱动	trace_block_rq_issue
C	io完成	trace_block_rq_complete
P	io plug	trace_block_plug
U	io unplup	trace_block_unplug
F	io前向合并	block_bio_backmerge
M	io后向合并	block_bio_frontmerge
X	io拆分	trace_block_split
R	io requeue	trace_block_rq_requeue

blktrace

io类型

- R 读
- W 写
- S 同步io
- D discard io
- M 元数据 io
- F flush io，刷磁盘缓存
- U orce unit access(fua)，数据要跳过缓存直接落盘

blktrace

blkprase解析

默认的输出格式是 %D %2c %8s %5T.%9t %5p %2a %3d

- %D 主次设备号
- %2c cpu号
- %8s 序列号
- %5T.%9t 时钟戳
- %2a action，代表io路径中的位置
- %3d io类型

最后根据action决定是否输出额外信息，一般有三个字段

- io 开始位置
- io 大小
- 进程名

实例1

大内存页同步小io

- 测试环境：64k page size
- 文件系统：xfs
- 测试参数：dd if=/dev/zero of=/mnt/test bs=4k count=1024 oflag=sync
- 存在问题：性能远小于4k page size场景
- 测试结果：
  - dd打印： 4194304 bytes (4.2 MB, 4.0 MiB) copied, 15.991 s, 262 kB/s
  - iostat数据

w/s	wMB/s	wrqm/s	%wrqm	w_await	aqu-sz	wareq-sz	%util
180.00	2.0	51.00	22.08	5.57	1.50	10.97	100.00

实例1

大内存页同步小io

- 测试结果
  - Blktrace数据

```
8,48  18     125     0.273247960  1150 C WS 2105616 + 8 [0]
8,48  18     132     0.290411700  1150 C WS 2105616 + 16 [0]
8,48  18     139     0.307578160  1150 C WS 2105616 + 24 [0]
8,48  18     146     0.324731900  1150 C WS 2105616 + 32 [0]
8,48  18     153     0.341879060  1150 C WS 2105616 + 40 [0]
8,48  18     160     0.359044200  1150 C WS 2105616 + 48 [0]
8,48  18     167     0.376196200  1150 C WS 2105616 + 56 [0]
8,48  18     174     0.393358220  1150 C WS 2105616 + 64 [0]
8,48  18     181     0.410508000  1150 C WS 2105616 + 72 [0]
8,48  18     188     0.427665960  1150 C WS 2105616 + 80 [0]
8,48  18     195     0.444819860  1150 C WS 2105616 + 88 [0]
8,48  18     202     0.461983320  1150 C WS 2105616 + 96 [0]
8,48  18     209     0.479134400  1150 C WS 2105616 + 104 [0]
8,48  18     216     0.496282220  1150 C WS 2105616 + 112 [0]
8,48  18     223     0.513446780  1150 C WS 2105616 + 120 [0]
8,48  18     230     0.530594240  1150 C WS 2105616 + 128 [0]
```

- 问题跟因：xfs不支持subpage，下发的io大小要大于4k

实例2

bfq并发同步io

- 测试环境：任意支持bfq调度器的环境(CONFIG_IOSCHED_BFQ)，并且开始基于bfq的io管控(CONFIG_BFQ_GROUP_IOSCHED)
- 测试方式：绑定进程到不同的 cgroup下，并发下同步io
- 存在问题：性能很差，与单进程性能持平
- 测试参数：每个进程：

```
[global]
filename=/dev/sdd
ioengine=psyncbs=4k
direct=1
numjobs=1
size=1g

[test]
rw=randwrite
```

OpenEuler

实例2

bfq并发同步io

- 测试结果
  - fio结果

进程数量	iops 每个进程	iops 总共
1	14k	14k
2	7k	14k
4	3.5k	14k

  - iostat

进程数量	w/s	wMB/s	wrqm/s	%wrqm	w_await	aqu-sz	wareq-sz	%util
1	15544.00	60.72	0.00	0.00	0.05	1.50	8.00	100.00
2	14976.00	58.50	0.00	0.00	0.12	1.81	8.00	100.00
4	15611.00	60.98	0.00	0.00	0.24	3.79	8.00	100.00

实例2

bfq并发同步io

- 测试结果
  - blktrace 多进程
    - io无法并发

```
8,0    5       99     0.001550075  1605 Q WS 992656 + 8 [fio]
8,0    5      100     0.001551290  1605 G WS 992656 + 8 [fio]
8,0    5      101     0.001551425  1605 P   N [fio]
8,0    5      102     0.001551627  1605 UT   N [fio] 1
8,0    5      103     0.001552049  1605 I   WS 992656 + 8 [fio]
8,0    5      104     0.001559766   427  D WS 992656 + 8 [kworker/5:1H]
8,0    5      105     0.001601695     0  C WS 992656 + 8 [0]
8,0    5      106     0.001612254  1605 Q WS 992664 + 8 [fio]
8,0    5      107     0.001613409  1605 G WS 992664 + 8 [fio]
8,0    5      108     0.001613545  1605 P   N [fio]
8,0    5      109     0.001613747  1605 UT   N [fio] 1
8,0    5      110     0.001614143  1605 I   WS 992664 + 8 [fio]
8,0    5      111     0.001622087   427  D WS 992664 + 8 [kworker/5:1H]
8,0    5      112     0.001662681     0  C WS 992664 + 8 [0]
8,0    5      113     0.001673206  1605 Q WS 992672 + 8 [fio]
8,0    5      114     0.001674380  1605 G WS 992672 + 8 [fio]
8,0    5      115     0.001674517  1605 P   N [fio]
8,0    5      116     0.001674733  1605 UT   N [fio] 1
8,0    5      117     0.001675175  1605 I   WS 992672 + 8 [fio]
8,0    5      118     0.001682875   427  D WS 992672 + 8 [kworker/5:1H]
8,0    5      119     0.001725369     0  C WS 992672 + 8 [0]
```

实例2

bfq并发同步io

- 测试结果
  - blktrace 多进程
    - D2C延时分布

```
Write latency
cnt 80406 sum 3718.9ms mean 0.0ms min 0.0ms max 6.2ms
>=(ms) .. <(ms)   : count     ratio |distribution                                    |
      0 .. 1       : 80405    100.0% |########################################|
      1 .. 2       : 0          0.0% |                                        |
      2 .. 4       : 0          0.0% |                                        |
      4 .. 8       : 1          0.0% |#                                       |
```

    - I2C延时分布

```
Write latency
cnt 80403 sum 20624.8ms mean 0.3ms min 0.0ms max 500.1ms
>=(ms) .. <(ms)   : count     ratio |distribution                                    |
      0 .. 1       : 80358     99.9% |########################################|
      1 .. 2       : 0          0.0% |                                        |
      2 .. 4       : 0          0.0% |                                        |
      4 .. 8       : 1          0.0% |#                                       |
      8 .. 16      : 0          0.0% |                                        |
     16 .. 32      : 0          0.0% |                                        |
     32 .. 64      : 0          0.0% |                                        |
     64 .. 128     : 0          0.0% |                                        |
    128 .. 256     : 3          0.0% |#                                       |
    256 .. 512     : 41         0.1% |#                                       |
```

OpenEuler

关注点

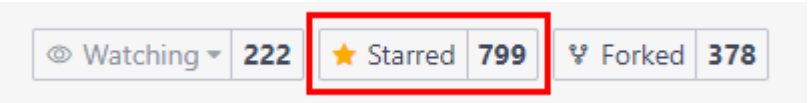
- io大小：buffer io时关注是否一致 (实例1)
- io延时: Q->G->I->D->C 各个阶段的延时情况 (实例2)
- io并发程度：同时处在D2C阶段的io数量，实例(2)
- io偏移量：机械盘需要关注
- io是否有合并：性能劣化可能是由于合并少导致的
- io是否拆分：io拆分后应该是顺序的io，关注完成顺序是否一致
- io类型

✓ openEuler kernel gitee 仓库

源代码仓库

https://gitee.com/openeuler/kernel

欢迎大家多多 Star，多多参与社区开发，多多贡献补丁。



✓ maillist、issue、bugzilla

可以通过邮件列表、issue、bugzilla 参与社区讨论

欢迎大家多多讨论问题，发现问题多提 issue、bugzilla

https://gitee.com/openeuler/kernel/issues

https://bugzilla.openeuler.org

kernel@openeuler.org

✓ openEuler kernel SIG 微信技术交流群

请扫描右方二维码添加小助手微信

或者直接添加小助手微信（微信号：openeuler-kernel）

备注“交流群”或“技术交流”

加入 openEuler kernel SIG 技术交流群

技术交流

OpenEuler

Thank you

OpenEuler