Linux任督二脉之内存管理(五)

讲解时间：6月10-14日晚9点

宋宝华 <21cnbao@gmail.com>

微信群直播

扫描二维码报名

麦当劳喜欢您来，喜欢您再来

扫描关注

Linux阅码场

其他工程问题以及调优

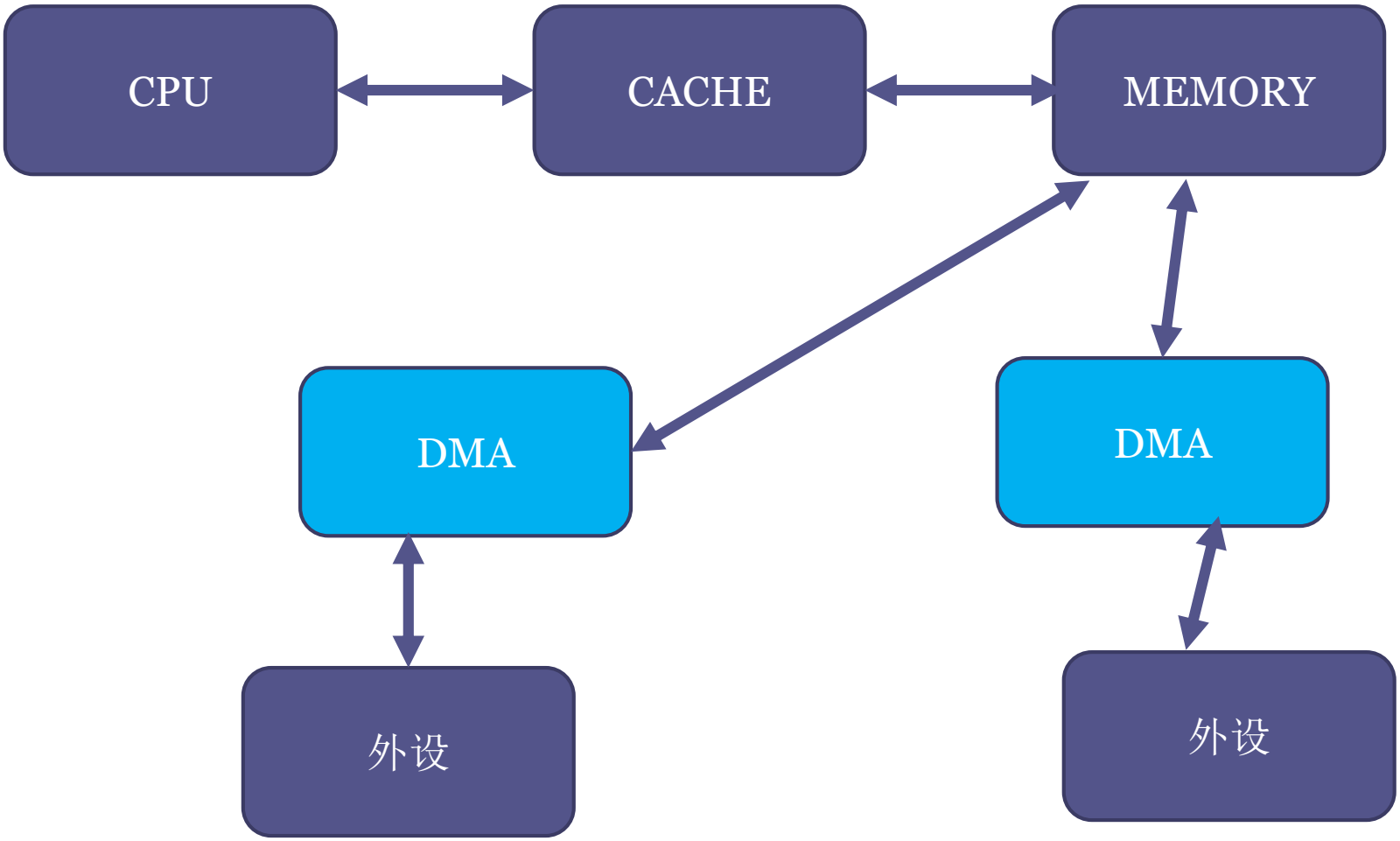
*DMA和cache一致性
*内存的cgroup
*性能方面的调优：page in/out, swapin/out
*Dirty ratio的一些设置
*swappiness

练习题

*vmstat;
*smem -s swap -t -k –n
*限制一个cgroup的memory，用cgexec把一个进程放到这个cgroup

DMA与Cache一致性

- DMA传输外设数据到memory，cache中可能是老的数据
- CPU写数据到memory，cache中是新数据，MEM是老数据



DMA APIs

- Coherent DMA buffers

```
void * dma_alloc_coherent(struct device *dev, size_t size, dma_addr_t *dma_handle,
gfp_t flag);

void dma_free_coherent (struct device *dev, size_t size,
void *cpu_addr, dma_addr_t dma_handle);
```

CMA和此API自动融合，调用dma_alloc_coherent()将从CMA获得内存。

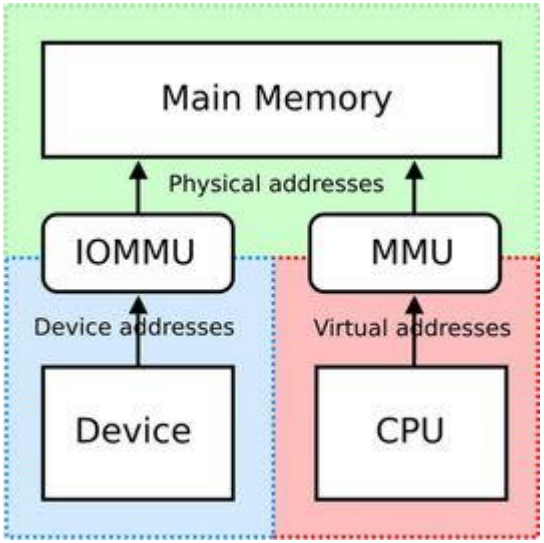
- DMA Streaming Mapping

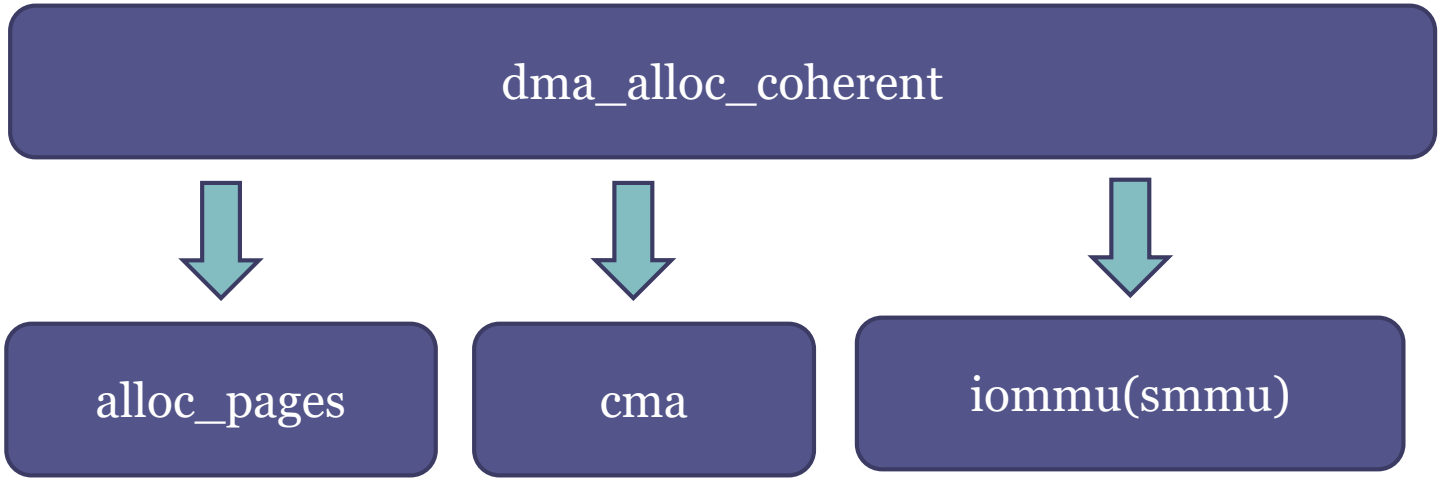
```
int dma_map_sg(...);
void dma_unmap_sg (...);
dma_addr_t dma_map_single(...);
void dma_unmap_single (...);
```

iommu(smmu)

- 把不连续的内存用来做DMA,以及限制DMA的访问范围（protection）
- IOMMU被融入到DMA APIs

I see no reason why ARM should be any different from other architectures which have IOMMUs, and I don't see why ARM should have to invent a whole new framework to handle IOMMUs. And I see no explanation why the existing hooks are unsuitable - at least as the _initial_ starting point.
-Russell King





cgroup

- 在Linux读写文件时，它用于缓存文件的逻辑内容，从而加快对磁盘上映像和数据的访问

设置cgroup A的最大内存为200MB

```
root@baohua-VirtualBox:~# cd /sys/fs/cgroup/memory/
root@baohua-VirtualBox:/sys/fs/cgroup/memory# mkdir A
root@baohua-VirtualBox:/sys/fs/cgroup/memory# cd A/
root@baohua-VirtualBox:/sys/fs/cgroup/memory/A# echo $((200*1024*1024)) > memory.limit_in_bytes
```

把进程放入cgroup A执行

```
root@baohua-VirtualBox:~/develop/training/memory-courses/day2# cgexec -g memory:A ./a.out
malloc buffer: 0x3a590008
0MB written
...
192MB written
196MB written
Killed
```

文件Dirty数据的写回

- dirty_ratio

the number of pages at which a process which is generating disk writes will itself start writing out dirty data.

- dirty_expire_centisecs

This tunable is used to define when dirty data is old enough to be eligible for writeout by the kernel flusher threads. It is expressed in 100'ths of a second. Data which has been dirty in-memory for longer than this interval will be written out next time a flusher thread wakes up.

- dirty_writeback_centisecs

The kernel flusher threads will periodically wake up and write `old' data out to disk. Setting this to zero disables periodic writeback altogether.

- dirty_background_ratio

the number of pages at which the background kernel flusher threads will start writing out dirty data.

vfs_cache_pressure

- 该文件表示内核回收用于directory和inode cache内存的倾向

This variable controls the tendency of the kernel to reclaim the memory which is used for caching of VFS caches, versus pagecache and swap.Increasing this value increases the rate at which VFS caches are reclaimed.It is difficult to know when this should be changed, other than by experimentation. The slabtop command (part of the package procps) shows top memory objects used by the kernel. The vfs caches are the "dentry"and the "*_inode_cache" objects. If these are consuming a large amount of memory in relation to pagecache, it may be worth trying to increase pressure. Could also help to reduce swapping. The default value is 100.

水位设置:min_free_kbytes与lowmem_reserve_ratio

- min_free_kbytes

This is used to force the Linux VM to keep a minimum number of kilobytes free. The VM uses this number to compute a watermark[WMARK_MIN] value for each lowmem zone in the system. Each lowmem zone gets a number of reserved free pages based proportionally on its size.

PF_MEMALLOC：紧急内存，可以忽略内存管理的水印进行分配

- ✓ **min_free_kbytes = 4 * sqrt(lowmem_kbytes), min_free_kbytes随着内存的增大不是线性增**
- ✓ watermark[min] = per_zone_min_free_pages
- ✓ watermark[high] - watermark[low] = watermark[low] - watermark[min] = per_zone_min_free_pages * 1/4

- lowmem_reserve_ratio

The `lowmem_reserve_ratio' tunable determines how aggressive the kernel is in defending these lower zones.

水位设置:high,low,min

```
/proc/sys/vm$ cat /proc/zoneinfo
Node 0, zone      DMA
  pages free     1228
        min      197
        low      246
        high     295
        ...
Node 0, zone   Normal
  pages free     21367
        min      10842
        low      13552
        high     16263
        ...
/proc/sys/vm$ cat min_free_kbytes
44160
```

high：
内存到此点，停止回收

low： 内存到此点，kswapd启动reclaim

Direct relaim:直接在应用程序的进程上下文中进行回收，会阻塞应用

min： 内存到此点，做direct reclaim

swappiness

- Swappiness反映是否积极地使用swap空间

✓swappiness = 0
仅在内存不足的情况下(free and file-backed pages < high water mark in a zone)，使用swap空间

✓swappiness = 60
默认值

✓swappiness = 100
内核将积极的使用swap空间。

```
root@baohua-VirtualBox:/proc/sys/vm# cat swappiness
60
```

getdelays

- Documentation/accounting/getdelays.c工具,测量调度、I/O、SWAP、Reclaim的延迟

```
root@baohua-VirtualBox:~/develop/linux/Documentation/accounting# gcc getdelays.c
root@baohua-VirtualBox:~/develop/linux/Documentation/accounting# ./a.out -dilv -p 1
print delayacct stats ON
printing IO accounting
listen forever
debug on
family id 25
Sent pid/tgid, retval 0
received 364 bytes
nlmsghdr size=16, nlmsg_len=364, rep_len=364
PID     1


CPU             count     real total  virtual total    delay total  delay average
                 3748     1692000000     1777525852      500993577          0.134ms
IO              count    delay total  delay average
                  194      116602567             0ms
SWAP            count    delay total  delay average
                    2         731594             0ms
RECLAIM         count    delay total  delay average
                    0              0             0ms
init: read=6844416, write=0, cancelled_write=0
^C
```

vmstat

- vmstat可以展现给定时间间隔的服务器的状态值,包括Linux的CPU使用率,内存使用,虚拟内存交换情况,IO读写情况。

```
baohua@baohua-VirtualBox:~$ vmstat 1
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 126444 361244  61208    1    4    43     5   23  101  2  0 98  0  0
 1  0      0 126440 361252  61212    0    0     0    16  990 4430 27  3 70  0  0
 1  0      0 125680 361252  61232    0    0     0     0  987 4753 28  2 70  0  0
 1  0      0 125204 361252  61248    0    0     0     0 1074 4533 28  2 69  0  0
 2  0      0 125292 361252  61260    0    0     0     0  998 4380 27  3 71  0  0
 1  0      0 125784 361252  61272    0    0     0     0  913 4915 28  2 70  0  0
 1  0      0 125844 361260  61276    0    0     0   332  995 4118 28  2 70  0  0
 2  0      0 124996 361260  61296    0    0     0     0  972 4415 26  3 71  0  0
 1  0      0 124992 361260  61308    0    0     0     0  978 5045 28  3 70  0  0
 2  0      0 125060 361260  61320    0    0     0     0  964 4229 27  2 71  0  0
 1  0      0 125072 361260  61332    0    0     0     0  993 5155 29  2 69  0  0
```

课程练习源码

https://github.com/21cnbao/memory-courses

课后阅读

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/performance_tuning_guide/s-memory-tunables

更早课程

- 《Linux总线、设备、驱动模型》录播：
http://edu.csdn.net/course/detail/5329

- 深入探究Linux的设备树
http://edu.csdn.net/course/detail/5627

- C语言大型软件设计的面向对象
https://edu.csdn.net/course/detail/6496

谢 谢！